Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques

19th International Workshop, APPROX 2016, and
20th International Workshop, RANDOM 2016
September 7–9, 2016, Paris, France

Edited by

Klaus Jansen
Claire Mathieu
José D. P. Rolim
Chris Umans

LIPICS

Editors

Klaus Jansen
University of Kiel
Kiel, Germany
`kj@informatik.uni-kiel.de`

Claire Mathieu
CNRS, Ecole Normale Superieure
Paris, France
`cmathieu@di.ens.fr`

Jośe Rolim
University of Geneva
Geneva, Switzerland
`Jose.Rolim@unige.ch`

Chris Umans
California Institute of Technology
Pasadena, CA, USA
`umans@cs.caltech.edu`

ACM Classification 1998
F.0 Theory of Computation, F.1.1 Models of Computation, F.1.2 Modes of Computation – Online Computation, F.1.3 Complexity Measures and Classes, F.2 Analysis of Algorithms and Problem Complexity, F.2.1 Numerical Algorithms and Problems – Computations on Matrices, F.2.2 Nonnumerical Algorithms and Problems – Computations on Discrete Structures – Sequencing and Scheduling – Geometrical Problems and Computations, F.2.3 Tradeoffs between Complexity Measures, G.1.3 Numerical Linear Algebra, G.2.1 Combinatorics – Counting Problems, G.2.2 [Discrete Mathematics] Graph Theory–Graph Algorithms, Path and Circuit Problems, G.3 Probability and Statistics – Probabilistic Algorithms, I.2.8 Problem Solving, Control Methods, and Search (Dynamic Programming)

ISBN 978-3-95977-018-7

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-95977-018-7.

Publication date
September, 2016

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 http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/LIPIcs.APPROX-RANDOM.2016.0

ISBN 978-3-95977-018-7 **ISSN 1868-8969** **http://www.dagstuhl.de/lipics**

ISSN 1868-8969

http://www.dagstuhl.de/lipics

Contents

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016).
Editors: Klaus Jansen, Claire Matthieu, José D. P. Rolim, and Chris Umans
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

This volume contains the papers presented at the 19th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2016) and the 20th International Workshop on Randomization and Computation (RANDOM 2016), which took place concurrently at the Institut Henri Poincaré in Paris, France during September 7–9, 2016.

APPROX focuses on algorithmic and complexity issues surrounding the development of efficient approximate solutions to computationally difficult problems, and was the 19th in the series after Aalborg (1998), Berkeley (1999), Saarbrücken (2000), Berkeley (2001), Rome (2002), Princeton (2003), Cambridge (2004), Berkeley (2005), Barcelona (2006), Princeton (2007), Boston (2008), Berkeley (2009), Barcelona (2010), Princeton (2011), Boston (2012), Berkeley (2013), Barcelona (2014), and Princeton (2015). RANDOM is concerned with applications of randomness to computational and combinatorial problems, and was the 20th workshop in the series following Bologna (1997), Barcelona (1998), Berkeley (1999), Geneva (2000), Berkeley (2001), Harvard (2002), Princeton (2003), Cambridge (2004), Berkeley (2005), Barcelona (2006), Princeton (2007), Boston (2008), Berkeley (2009), Barcelona (2010), Princeton (2011), Boston (2012), Berkeley (2013), Barcelona (2014), and Princeton (2015).

Topics of interest for APPROX and RANDOM are: design and analysis of approximation algorithms, hardness of approximation, small space algorithms, sub-linear time algorithms, streaming algorithms, embeddings and metric space methods, spectral methods, mathematical programming methods, combinatorial optimization in graphs and networks, algorithmic game theory, mechanism design and economics, computational geometric problems, distributed and parallel approximation, approximate learning, online algorithms, approaches that go beyond worst case analysis, 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, property testing, computational learning theory, and other applications of approximation and randomness.

The volume contains 20 contributed papers, selected by the APPROX Program Committee out of 40 submissions, and 27 contributed papers, selected by the RANDOM Program Committee out of 45 submissions.

We would like to thank all of the authors who submitted papers, the invited speakers, the members of the Program Committees, and the external reviewers. We gratefully acknowledge the Institute of Computer Science of the Christian-Albrechts-Universität zu Kiel, the Computer Science Department of École normale Supérieure, the Department of Computer Science of the University of Geneva, and the Computing and Mathematical Sciences Department at Caltech.

September 2016

Klaus Jansen, Claire Matthieu
José D. P. Rolim, and Chris Umans

Program Committees

APPROX 2016

Anna Adamaszek	University of Copenhagen, Denmark
Shiri Chechik	Tel Aviv, Israel
Anne Driemel	TU Eindhoven, The Netherlands
Lee-Ad Gottlieb	Ariel University, USA
Varun Kanade	University of Oxford, United Kingdom
Nitish Korula	Google Research, USA
Stefano Leonardi	Sapienza University of Rome, Italy
Daniel Lokshtanov	University of Bergen, Norway
Claire Mathieu (chair)	École normale supérieure, France
Nicole Megow	Technische Universität München, Germany
Tobias Moemke	Universität des Saarlandes, Germany
Shayan Oveis Gharan	University of Washington, USA
Debmalya Panigrahi	Duke University, USA
Richard Peng	Georgia Institute of Technology, USA
Ely Porat	Bar-Ilan University, Israel
Adi Rosén	Université Paris Diderot - Paris 7, France
Adrian Vetta	McGill University, Canada
Rico Zenklusen	ETH Zurich, Switzerland

RANDOM 2016

Mahdi Cheraghchi	Imperial College London, United Kingdom
Elena Grigorescu	Purdue University, USA
Neeraj Kayal	Microsoft Research, India
Adam Klivans	University of Texas, USA
Swastik Kopparty	Rutgers University, USA
Ravi Kumar	Google Research, USA
Dana Moshkovitz	Massachusetts Institute of Technology, USA
Ashwin Nayak	University of Waterloo, Canada
Ryan O'Donnell	Carnegie Mellon University, USA
Asaf Shapira	Tel-Aviv University, Israel
Ronen Shaltiel	University of Haifa, Israel
Alexander Sherstov	University of California, USA
Thomas Thierauf	Aalen University, Germany
Chris Umans (chair)	California Institute of Technology, USA
Eric Vigoda	Georgia Institute of Technology, USA

External Reviewers

Dimitris Achlioptas
David Adjiashvili
Alex Andoni
Antonios Antoniadis
Per Austrin
Nikhil Bansal
Mohammadhossein Bateni
Louay Bazzi
Itai Benjamini
Mikhail Berlinkov
Ivona Bezakova
Prateek Bhakta
Amey Bhangale
Arnab Bhattacharyya
Vijay Bhattiprolu
Andrej Bogdanov
Zvika Brakerski
Michael Brautbar
Joshua Brody
Eshan Chattopadhyay
Dehua Cheng
Steve Chestnut
Flavio Chierichetti
Vincent Cohen-Addad
Ben Cousins
Anirban Dasgupta
Stephen Desalvo
Carola Doerr
Charilaos Efthymiou
Leah Epstein
Hossein Esfandiari
Uriel Feige
Jan Foniok
Lance Fortnow
Alan Frieze
Takuro Fukunaga
Andreas Galanis
Ankit Garg
Rong Ge
Parikshit Gopalan
Martin Groß
Gus Gutoski
Samuel Haney
Thomas Dueholm Hansen
Pooya Hatami
Lisa Hellerstein
Sangxia Huang
T.S. Jayram
Hossein Jowhari
Michael Kapralov
Nathaniel Kell
David Kempe
Guy Kindler
Philip Klein
Gillat Kol
Christian Konrad
Pravesh Kothari
Ioannis Koutis
Euiwoong Lee
Christoph Lenzen
Amit Levi
Vahid Liaghat
Laci Lovasz
Shachar Lovett
Jannik Matuschke
Andrew McGregor
Manor Mendel
Sarah Miracle
Neeldhara Misra
Viswanath Nagarajan
Jelani Nelson
Sasho Nikolov
Sergey Norin
Zeev Nutov
Igor Pak
Konstantinos Panagiotou
Daniel Paulusma
Chris Peikert
Pan Peng
Sofya Raskhodnikova
Dana Ron
Noga Ron-Zewi
Tselil Schramm
Igor Shinkar
Amir Shpilka
Adam Smith
Shay Solomon
Frits Spieksma
Daniel Spielman
Nikhil Srivastava

Daniel Stefankovic
Tibor Szabo
Justin Thaler
Charalampos Tsourakakis
Erik Jan van Leeuwen
Rob van Stee
Stephen Vavasis
Mikhail Volkov
Thomas Watson
Weiqiang Wen
Andreas Wiese
Anthony Wirth
David Woodruff
Yitong Yin
Samson Zhou
David Zuckerman

List of Authors

Jasine Babu
Arturs Backurs
Victor Bapst
Amitabh Basu
Arnab Bhattacharyya
Abhishek Bhowmick
Ivan Bliznets
Ravi Boppana
Vladimir Braverman

Guillaume Chapuy
Moses Charikar
Lin Chen
Eden Chlamtáč
Michael B. Cohen
Amin Coja-Oghlan
Marek Cygan

Daniel Dadush
Michael Dinitz

Esther Ezra

Uriel Feige
Michal Feldman

Bernd Gärtner
Shashwat Garg
Heng Guo
Chetan Gupta

Nir Halman
Prahladh Harsha
Johan Håstad
Pooya Hatami
Jan Hązła
Thomas Holenstein
Carlos Hoppen

Sungjin Im

Varun Kanade
Subhash Khot
Areej Khoury
Anthony Kim
Yoshiharu Kohayakawa
Paweł Komosa
Christian Konrad
Guy Kortsarz
Samir Khuller Janardhan Kulkarni

Richard Lang
Chin Ho Lee
Hanno Lefmann
Nikos Leonardos
Reut Levi
Xin Li
Yi Li
Vahid Liaghat
Shachar Lovett
Pinyan Lu

Frédéric Magniez
Konstantin Makarychev
Yury Makarychev
Pasin Manurangsi
Dániel Marx
Andrew McGregor
Dhruv Medarametla
Ryuhei Mori
Benjamin Moseley
Dana Moshkovitz
Elchanan Mossel
Kamesh Munagala
Cameron Musco

Yonatan Naamad
Preetum Nakkiran
Ilan Newman
Cyril Nicaud
Aleksandar Nikolov
Zeev Nutov

Jakub Pachocki
Guillem Perarnau
Will Perkins
Michał Pilipczuk
Aaron Potechin

Junjie Qin

George Rabanca
Sharath Raghvendra
Govind Ramnarayan
Anup Rao
Dana Ron
Alan Roytman
Ronitt Rubinfeld

Amin Saberi
Ario Salmasi
Ronen Shaltiel
Igor Shinkar
Anastasios Sidiropoulos
Jad Silbak
Makrand Sinha
Renjie Song
Srikanth Srinivasan
Henrique Stagni
Noah Stephens-Davidowitz
Maxim Sviridenko
Ridwan Syed

Inbal Talgam-Cohen
Antonis Thomas
Luca Trevisan
Madhur Tulsiani

Emanuele Viola
Sofya Vorotnikova
Gregory Vorsanger

Justin Ward
Anthony Wirth
David Witmer
David P. Woodruff

Sheng Yang
Deshi Ye
Yitong Yin
Henry Yuen

Chihao Zhang
Guochuan Zhang
Jinman Zhao

Constant-Distortion Embeddings of Hausdorff Metrics into Constant-Dimensional ℓ_p Spaces

Artūrs Bačkurs[*1] and **Anastasios Sidiropoulos**[†2]

1 **MIT, Cambridge MA, USA**
backurs@mit.edu

2 **The Ohio State University, Columbus OH, USA**
sidiropoulos.1@osu.edu

Abstract

We show that the Hausdorff metric over constant-size pointsets in constant-dimensional Euclidean space admits an embedding into constant-dimensional ℓ_∞ space with constant distortion. More specifically for any $s, d \geq 1$, we obtain an embedding of the Hausdorff metric over pointsets of size s in d-dimensional Euclidean space, into $\ell_\infty^{s^{O(s+d)}}$ with distortion $s^{O(s+d)}$. We remark that any metric space M admits an isometric embedding into ℓ_∞ with dimension proportional to the size of M. In contrast, we obtain an embedding of a space of infinite size into constant-dimensional ℓ_∞.

We further improve the distortion and dimension trade-offs by considering probabilistic embeddings of the snowflake version of the Hausdorff metric. For the case of pointsets of size s in the real line of bounded resolution, we obtain a probabilistic embedding into $\ell_1^{O(s \log s)}$ with distortion $O(s)$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases metric embeddings, Hausdorff metric, distortion, dimension

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.1

1 Introduction

Low-distortion embeddings between metric spaces have given rise to a plethora of tools in computer science and mathematics [10, 15, 2, 5, 17]. The most well-studied case is embedding into ℓ_p^d, that is $\mathbb{R}^d$ endowed with the ℓ_p distance. In this case the most important parameters are the distortion of the embedding and the dimension of the target space; the former quantifies the extent to which the geometry of the input space is preserved, while the latter affects the complexity of various algorithmic methods performed on the target space.

In most embeddings of finite metric spaces both of these parameters depend on the size of the input space. Prototypical such examples are Bourgain's Theorem [4] which asserts that any n-point metric admits an embedding into ℓ_2 with distortion $O(\log n)$, and the seminal result of Johnson and Lindenstrauss [14] asserting that any n-point subset of ℓ_2 admits an embedding into $\ell_2^{O(\varepsilon^{-2} \log n)}$ with distortion $1 + \varepsilon$.

However, in several applications one seeks an embedding of some input space of infinite size. One such application is in algorithms and data structures (e.g. nearest neighbor data structures) with approximation guarantee independent of the input size. Another application

[*] Arturs Backurs was supported by NSF and Simons Foundation.
[†] Anastasios Sidiropoulos is supported by NSF under grant CCF-1423230 and award CAREER-1453472.

is when designing an oblivious or streaming algorithm that requires an embedding of the input space that can be computed independently at each point without having access to the rest of the input (e.g. [12, 13]).

A classical example of an embedding of an infinite metric is Dvoretzky's Theorem [6] which asserts that for any $k \geq 1$, there exists $d \geq 1$ such that ℓ_2^k admits an embedding into any d-dimensional normed space with distortion $1 + \varepsilon$.

Interestingly, the case of input spaces that are not normed, is much less understood. One important such space is given by the Hausdorff metric which is used to measure the dissimilarity between two pointsets. Given two finite pointsets $A = \{a_1, \dots, a_s\}$ and $B = \{b_1, \dots, b_s\}$, the Hausdorff distance is defined as

$$H_s(A, B) = \max(h(A, B), h(B, A)),$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} m(a, b)$$

and $m(\cdot, \cdot)$ is underlying metric on the points of A and B. We use the notation $H_{s,d}$ to denote the Hausdorff distance with underlying metric ℓ_2^d. We will omit subscripts if the cardinality of the pointsets or the underlying metric is clear from the context, or the statement is valid independent from the cardinality or the underlying space.

We study embeddings of the Hausdorff metric over finite subsets of Euclidean space. This is an infinite space since there are infinitely many possible subsets even in the real line. Therefore known results for embedding finite metrics into ℓ_p space are not directly applicable in this case.

1.1 Our results and techniques

1.1.1 Embedding for Hausdorff metric over pointsets in $\mathbb{R}^1$

We show that there exists an embedding of $H_{s,1}$ into $\ell_\infty^{s^{O(s)}}$ with distortion $s^{O(s)}$. Let $\mathcal{M}$ be a collection of metric spaces on the same pointset X. We say that a metric is a *ℓ_∞-metric over $\mathcal{M}$* if for any pair of points in X the distance is given by the maximum over all distances in $\mathcal{M}$. Our result is obtained via iteratively embedding H_s into an ℓ_∞-metric over H_{s-1} metrics. The key property in this mapping is that it preserves all distances in the infinite space H_s. Repeating this process we inductively obtain an embedding of H_s into an ℓ_∞-metric over H_1 metrics. Since $H_1 = \mathbb{R}^1$, the resulting embedding is into ℓ_∞.

1.1.2 Embedding for Hausdorff metric over pointsets in $\mathbb{R}^d$

We extend the above approach to $H_{s,d}$. This is done by embedding $H_{s,d}$ into an ℓ_∞-metric over $H_{s,d-1}$ metrics. By repeating this embedding we obtain an embedding of $H_{s,d}$ into an ℓ_∞ metric over $H_{s,1}$ metrics. Combining with the above embedding we obtain the desired embedding of $H_{s,d}$ into ℓ_∞.

1.1.3 Probabilistic embeddings

The above embeddings obtain distortion and dimension that depend only on s. We show how to exponentially improve the dependence of both parameters on s by considering probabilistic embeddings of the snowflake version of H_s into ℓ_1.

Table 1 Summary of our and previous results on embedding Hausdorff distance into ℓ_p spaces. Column "dimension" specifies the dimension of the target ℓ_p space. We consider Hausdorff distance over pointsets of size s coming from the underlying space. Here, $\varepsilon > 0$ is a small constant.

	Underlying space	To	Dimension	Distortion	Comments
Theorem 10	ℓ_2^d	ℓ_∞	$s^{O(s+d)}$	$s^{O(s+d)}$	
Theorem 23	ℓ_p^1	ℓ_1	$O(s \log s)$	$O(s/\alpha)$	Snowflaked embedding with parameter α
[11]	$\{0,\dots,\Delta\}_\infty^d$	ℓ_∞	$s^2 \cdot e^{O(d)} \cdot \log^2 \Delta$	$1+\varepsilon$	Threshold embedding
[16]	ℓ_2^d	ℓ_∞	$e^{O(ds)}$	$1+\varepsilon$	Snowflaking

1.1.4 Embedding into high-dimensional ℓ_1 space

To improve the distortion of the embedding, we relax the requirements of the embedding. First, we embed a snowflake version of the Hausdorff distance into ℓ_1. This means that we embed the distance $H_s^{1-\alpha}$ for some $\alpha > 0$ into ℓ_1. Second, we allow that the expansion property holds in expectation (see Section 2 for a formal definition). This allows us to achieve distortion $O(s/\alpha)$, which is an exponential improvement over the deterministic embedding. The embedding uses ideas that were previously used to construct embeddings for earth-mover distance [9, 1]. In particular, we recursively subdivide the underlying metric space into cells and designate a coordinate in the target space for every cell. Instead of counting the number of points that fall into each cell (as was done in the case of embeddings of earth-mover distance), we instead detect whether at least one point falls into the cell. To achieve distortion that does not depend on the size of the underlying metric, we use ideas developed in [1], embedding a snowflake version of the Hausdorff distance.

1.1.5 Embedding into low-dimensional ℓ_1 space

To improve the dimension of the target ℓ_1 space, we further relax the requirements of the embedding. We allow that the embedding contracts with probability bounded by some small constant. This allows us to reduce the dimension exponentially. The dimension of the target ℓ_1 space becomes $O(s \log s)$. This improvement is obtained by observing that a vector resulting from the embedding into high dimensional space, is essentially sparse; that is, the main contribution to the ℓ_1 norm comes from few non-zero entries. This suggests that we can use dimensionality reduction techniques for ℓ_1 space for sparse vectors. To this end we use a construction from [3]. We remark that a similar dimensionality reduction idea was used in [1]. We note that we can decrease the probability of contraction to an arbitrary $\delta > 0$ by combining $O(\log(1/\delta))$ independent copies of the embedding.

In Table 1 we summarize our results and highlight the previous work on embedding Hausdorff distance into simpler spaces.

1.2 Related work

Farach-Colton and Indyk [7] have studied the problem of embedding the Hausdorff metric over finite pointsets into ℓ_∞. However, they only obtain embeddings that approximately preserve distances that are within a fixed range $[r, R]$, for some $0 < r < R$. This weaker guarantee is sufficient for designing an approximate nearest neighbor data structure. However, in order to obtain an embedding that preserves all distances up to some small distortion, one has to concatenate $O(\log \Delta)$ such embeddings, where Δ is the spread of the metric. Since Δ is in general unbounded, this leads to a host space of arbitrarily large dimension.

Indyk [11] studied *threshold embeddings* for Hausdorff distance. In this setting the goal is to obtain an embedding so that the following two conditions hold: First, the embedding is a contraction. Second, if the distance between two points in the original space is at least r, then their distance in the target space is at least r', for some $r \geq r' > 0$. The distortion of a threshold embedding is defined to be the ratio r/r'. The dimension of the target space in [11] depends on the size of the underlying metric, which can be unbounded.

Previous works [8, 16] studied embeddings of snowflake metrics. They showed that, if the *doubling dimension* of a metric is t, then it is possible to embed such a metric into $\ell_\infty^{e^{O(t)}}$ with distortion $1+\epsilon$, for any constant $\epsilon > 0$. We will not define doubling dimension here but we note that for the case of Hausdorff metric over ℓ_p^d, it is bounded by $O(ds)$.

2 Preliminaries

▶ **Definition 1.** Consider the Hausdorff distance over pointsets in some underlying space. Let f be a function that maps pointsets to vectors in some ℓ_∞-space. We say that f is an *embedding* if there exist $L \geq l > 0$ such that

$$l \cdot \|f(A) - f(B)\|_\infty \leq H(A,B) \leq L \cdot \|f(A) - f(B)\|_\infty \tag{1}$$

for all pointsets A and B. The quantity L/l is the *distortion* of the embedding.

▶ **Definition 2.** Let $\mathcal{D}$ be a probability distribution over functions that map pointsets of some space into some ℓ_∞-space. We say that a function f chosen from $\mathcal{D}$ is a *probabilistic embedding*. Moreover, if there exist $L \geq l > 0$ such that for all sets A, B, we have

$$l \cdot E_f[\|f(A) - f(B)\|_\infty] \leq H(A,B)$$

and

$$\Pr_f\left[H(A,B) \leq L \cdot \|f(A) - f(B)\|_\infty\right] \geq 2/3,$$

then the *distortion* of f is defined to be L/l. Note that the choice of $2/3$ is arbitrary; we can amplify it by sampling independent copies of the function f and concatenating the resulting embeddings.

▶ **Definition 3.** A probabilistic embedding f is called a *snowflaked embedding* with parameter $\alpha > 0$ if it satisfies the following properties: There exist $L \geq l > 0$ such that for all sets A, B, we have

$$l \cdot E_f[\|f(A) - f(B)\|_\infty] \leq H^{1-\alpha}(A,B)$$

and

$$\Pr_f\left[H^{1-\alpha}(A,B) \leq L \cdot \|f(A) - f(B)\|_\infty\right] \geq 2/3.$$

The *distortion* of f is defined to be L/l.

2.1 Notation

Given two vectors $A \in \mathbb{R}^x$, $A = (a_1, \ldots a_x)^T$ and $B \in \mathbb{R}^y$, $B = (b_1, \ldots, b_y)^T$, we denote concatenation of A and B by

$$A \oplus B := (a_1, \ldots, a_x, b_1, \ldots, b_y)^T.$$

For an integer n, we denote the set $\{1, 2, \ldots, n\}$ by $[n]$. For any $x, y \in \mathbb{R}$, we denote the set $\{x \leq z \leq y \mid z \in \mathbb{R}\}$ by $[x, y]$. For any $X \subseteq \mathbb{R}$ and $y \in \mathbb{R}$, we denote the set $\{x \cdot y \mid x \in X\}$ by $X \cdot y$. Similarly, we denote the set $\{x - y \mid x \in X\}$ by $X - y$. For any function $g : \mathbb{R} \to \mathbb{R}$ and $X \subseteq \mathbb{R}$, we denote the set $\{g(x) \mid x \in X\}$ by $g(X)$.

3 Embedding for Hausdorff metric over pointsets in $\mathbb{R}^1$

Below we will work with $H_{s,1}(A, B)$ and we will write $H_s(A, B)$ instead of $H_{s,1}(A, B)$.

We define $10s^2$ functions $f^i : [0, 1] \to [0, 1]$, one for each $i \in \{1, \ldots, 10s^2\}$, as follows.

$$f^i(x) := \begin{cases} \frac{x}{y_i} & \text{if } x \leq y_i; \\ \frac{1-x}{1-y_i} & \text{otherwise,} \end{cases}$$

where $y_i = \frac{1}{3} + \frac{i}{3(10s^2+1)}$. Notice that the function f^i satisfies the following four properties:

1. f^i achieves the maximum value 1 at y_i;
2. $\frac{1}{3} < y_i < \frac{2}{3}$;
3. $f^i(0) = f^i(1) = 0$;
4. f^i grows linearly in the interval $[0, y_i]$ and decreases linearly in the interval $[y_i, 1]$.

To prove our results, we need the following lemma.

▶ **Lemma 4.** *Let $A = \{a_1, \ldots, a_s\} \subseteq \mathbb{R}$ and $B = \{b_1, \ldots, b_s\} \subseteq \mathbb{R}$ with $0 = a_1 \leq a_2 \leq \ldots \leq a_s = 1$ and $0 = b_1 \leq b_2 \leq \ldots \leq b_s = 1$. We have*

$$H_s(A, B) \in [1/10, 1000s^2] \cdot \max_i H_{s-1}\left(f^i(A), f^i(B)\right).$$

Proof. Notice that, since $f^i(0) = f^i(1) = 0$, we have $|f^i(A)|, |f^i(B)| = s - 1$. This is why we have H_{s-1} in the right side of the equation in the statement of the lemma. From now on we will use H instead of H_s or H_{s-1}.

Now we will establish $H(A, B) \geq \frac{1}{10} \max_i(H(f^i(A), f^i(B)))$. It is sufficient to show that $H(A, B) \geq \frac{1}{10} H(f^i(A), f^i(B))$ for all $i \in [10s^2]$. Fix $i \in [10s^2]$. We can check that for all $x, y \in [0, 1]$, $|f^i(x) - f^i(y)| \leq 10|x - y|$ (f^i is a piece-wise linear function with the derivative bounded by 3 in absolute value in every piece). That is, f^i is a Lipschitz function with constant 10 in the interval $[0, 1]$. That means that f^i can increase distance between any two points by a factor of at most 10. Therefore, inequality $H(A, B) \geq \frac{1}{10} H(f^i(A), f^i(B))$ follows.

It remains to show that $H(A, B) \leq 1000s^2 \max_i H_{s-1}\left(f^i(A), f^i(B)\right)$. The remainder of the proof is devoted to show this inequality. We need to show that there exists $i \in [10s^2]$ such that $H(f^i(A), f^i(B)) \geq \frac{1}{1000s^2} H(A, B)$. We will show that there exists i such that

$$\forall a \in A, \quad d(f^i(a), f^i(B)) \geq \frac{1}{1000s^2} d(a, B) \tag{2}$$

and

$$\forall b \in B, \quad d(f^i(b), f^i(A)) \geq \frac{1}{1000s^2} d(b, A), \tag{3}$$

where function d is defined as follows. For point y and finite pointset X,

$$d(y, X) := \min_{x \in X} \|x - y\|.$$

The first inequality shows that for every point from A, the distance to the closest point from B decreases by a factor of at most $1000s^2$ if we apply map f^i to the point and to the set B. Similarly, the second inequality shows that for every point from B, the distance to the closest point from A decreases by a factor of at most $1000s^2$ if we apply map f^i to the point and to the set A. By the definition of Hausdorff distance, it is sufficient to show these two inequalities to establish what we need.

To prove (2) and (3), we will use the following proposition.

▶ **Proposition 5.** *For all $i_1 \neq i_2$ and $x, y \in [0, 1]$ with $x \leq y$,*

$$\frac{1}{1000s^2} \min(x, y - x) \leq \max\left(|f^{i_1}(x) - f^{i_1}(y)|, |f^{i_2}(x) - f^{i_2}(y)|\right) \tag{4}$$

and

$$\frac{1}{1000s^2} \min(1 - y, y - x) \leq \max\left(|f^{i_1}(x) - f^{i_1}(y)|, |f^{i_2}(x) - f^{i_2}(y)|\right). \tag{5}$$

Proof. We will show (4). The proof of (5) is analogous. W.l.o.g., $i_1 < i_2$. We have that $y_{i_1} < y_{i_2}$ (see the definition of function f^i). If $x \geq y_{i_1}$, we have that $|f^{i_1}(x) - f^{i_1}(y)| \geq y - x$ by the definition of the function f^{i_1}. Similarly, if $y \leq y_{i_2}$, we have that $|f^{i_2}(x) - f^{i_2}(y)| \geq y - x$ by the definition of function f^{i_2}. Therefore, if $x \geq y_{i_1}$ or $y \leq y_{i_2}$,

$$y - x \leq \max\left(|f^{i_1}(x) - f^{i_1}(y)|, |f^{i_2}(x) - f^{i_2}(y)|\right) \tag{6}$$

and we are done proving (4) and (5).

Now we consider the complement case: $x \leq y_{i_1}$ and $y \geq y_{i_2}$. We will show inequality

$$\frac{1}{1000s^2} x \leq \max\left(|f^{i_1}(x) - f^{i_1}(y)|, |f^{i_2}(x) - f^{i_2}(y)|\right). \tag{7}$$

Notice that, by combining (6) and (7), we get (4). Suppose that

$$|f^{i_1}(x) - f^{i_1}(y)| < \frac{1}{1000s^2} x \tag{8}$$

since otherwise we have established (7). By the definition of f^{i_1} and f^{i_2},

$$f^{i_1}(x) - f^{i_2}(x) = 3x \cdot \left(\frac{1}{1 + \frac{i_1}{10s^2+1}} - \frac{1}{1 + \frac{i_2}{10s^2+1}}\right) \geq \frac{x}{20s^2}, \tag{9}$$

where we use the fact that $i_2 - i_1 \geq 1$. Using inequalities $f^{i_2}(y) \geq f^{i_1} y$, (8) and (9), we get

$$\begin{aligned} |f^{i_2}(x) - f^{i_2}(y)| &\geq \left(f^{i_2}(y) - f^{i_1}(y)\right) + \left(f^{i_1}(x) - f^{i_2}(x)\right) - |f^{i_1}(y) - f^{i_1}(x)| \\ &\geq \frac{x}{20s^2} - \frac{x}{1000s^2} \geq \frac{x}{1000s^2}. \end{aligned}$$

This establishes (7). ◀

Now we continue the proof of Lemma 4. We will use several times the fact that $\{0, 1\} \subseteq A, B$. Consider $a \in A$ and $b \in B$ with $a \leq b$. By (4), inequality

$$\frac{1}{1000s^2} d(a, B) \leq \frac{1}{1000s^2} \min(a, b - a) \leq |f^i(a) - f^i(b)| \tag{10}$$

holds for all indices $i \in [10s^2]$ except at most one. Consider $a \in A$ and $b \in B$ with $a > b$. By (5), inequality

$$\frac{1}{1000s^2} d(a,B) \leq \frac{1}{1000s^2} \min(1-a, a-b) \leq |f^i(a) - f^i(b)| \tag{11}$$

holds for all indices $i \in [10s^2]$ except at most one. By fixing $a \in A$ and considering all $b \in B$, from (10) and (11) we have that

$$\frac{1}{1000s^2} d(a,B) \leq d(f^i(a), f^i(B)) \tag{12}$$

holds for all indices $i \in [10s^2]$ except for at most s indices.

Analogously we get that for any fixed $b \in B$,

$$\frac{1}{1000s^2} d(b,A) \leq d(f^i(b), f^i(A)) \tag{13}$$

holds for all indices $i \in [10s^2]$ except for at most s indices.

From (12) we get that (2) holds for all but s^2 indices. From (13) we get that (3) holds for all but s^2 indices. By definition of f^i, we consider $10s^2$ indices. We conclude that there must be at least $10s^2 - 2s^2 \geq 1$ index that satisfy both (2) and (3). This concludes the proof of the lemma. ◀

We define function $g^i_t(x) : \mathbb{R} \to \mathbb{R}$, parameterized by $t \in \mathbb{R}$ and $i \in [10s^2]$, as follows:

$$g^i_t(x) := t \cdot f^i\left(\frac{x}{t}\right).$$

▶ **Lemma 6.** *Let $A = \{a_1, \ldots, a_s\} \subseteq \mathbb{R}$ and $B = \{b_1, \ldots, b_s\} \subseteq \mathbb{R}$ with $a_1 \leq \ldots \leq a_s$ and $b_1 \leq \ldots \leq b_s$, and $a_1 = b_1$, and $a_s = b_s$. We have*

$$H_s(A,B) \in [1/10, 1000s^2] \cdot \max_i H_{s-1}\left(g^i_{a_s - a_1}(A - a_1), g^i_{b_s - b_1}(B - b_1)\right).$$

Proof. Hausdorff distance is shift invariant, that is, for any $x \in \mathbb{R}$, $H(A,B) = H(A-x, B-x)$. Because of this and $a_1 = b_1$, we can assume that $a_1 = b_1 = 0$. Then the inequality we want to prove simplifies to

$$H_s(A,B) \in [1/10, 1000s^2] \cdot \max_i H_{s-1}\left(g^i_{a_s}(A), g^i_{b_s}(B)\right). \tag{14}$$

By the definition of Hausdorff distance, for any positive $y \in \mathbb{R}$, $H(A,B) = \frac{H(x \cdot A, x \cdot B)}{x}$. Because of this equality, expression (14) follows from Lemma 4 and the definition of g^i_t and f^i. ◀

▶ **Lemma 7.** *Let $A = \{a_1, \ldots, a_s\} \subseteq \mathbb{R}$ and $B = \{b_1, \ldots, b_s\} \subseteq \mathbb{R}$ with $a_1 \leq \ldots \leq a_s$ and $b_1 \leq \ldots \leq b_s$. We have*

$$\begin{aligned} H_s(A,B) \in & [1/1000, 10^6 s^2] \\ & \cdot \max\left(|a_1 - b_1|, |a_s - b_s|, \max_i H_{s-1}\left(g^i_{a_s - a_1}(A - a_1), g^i_{b_s - b_1}(B - b_1)\right)\right). \end{aligned}$$

Proof. We define pointsets $A' = \{a'_1, \ldots, a'_s\}$ and $B' = \{b'_1, \ldots, b'_s\}$ from A and B in the following way.

1. We set $a'_i = a_i$ and $b'_i = b_i$ for all $i \in [s]$;
2. if $a'_1 < b'_1$, we set b'_1 to be equal to a'_1;

3. if $a'_1 > b'_1$, we set a'_1 to be equal to b'_1;
4. if $a'_s < b'_s$, we set a'_s to be equal to b'_s;
5. if $a'_s > b'_s$, we set b'_s to be equal to a'_s.

We define $M := \max(|a_1 - b_1|, |a_s - b_s|)$.

▶ **Proposition 8.**

$$\left|H\left(g^i_{a_s-a_1}(A-a_1), g^i_{b_s-b_1}(B-b_1)\right) - H\left(g^i_{a'_s-a'_1}(A'-a'_1), g^i_{b'_s-b'_1}(B'-b'_1)\right)\right| \le 100M.$$

Proof. Notice that for every x, such that $0 \le x \le a_s - a_1$,

$$|g^i_{a_s-a_1}(x) - g^i_{a'_s-a'_1}(x)| \le 50M. \tag{15}$$

This is true because g^i is a Lipshitz function with Lipshitz constant at most 10 and $|a_s - a'_s|, |a_1 - a'_1| \le M$. Similarly, for every x, $0 \le x \le b_s - b_1$,

$$|g^i_{b_s-b_1}(x) - g^i_{b'_s-b'_1}(x)| \le 50M. \tag{16}$$

(15) and (16) mean that, as we apply function g^j to set A' instead of A and to B' instead of B, every point in the resulting sets (after application of g^j) changes its position by at most $50M$, and the assertion follows. ◀

By Lemma 6,

$$H(A', B') \in [1/10, 1000s^2] \cdot \max_i H\left(g^i_{a'_s-a'_1}(A'-a'_1), g^i_{b'_s-b'_1}(B'-b'_1)\right).$$

We get

$$\begin{aligned}
&\max\left(|a_1 - b_1|, |a_s - b_s|, \max_i H\left(g^i_{a_s-a_1}(A-a_1), g^i_{b_s-b_1}(B-b_1)\right)\right)\\
\le& M + \max_i H\left(g^i_{a_s-a_1}(A-a_1), g^i_{b_s-b_1}(B-b_1)\right)\\
\le& M + 100M + \max_i H\left(g^i_{a'_s-a'_1}(A'-a'_1), g^i_{b'_s-b'_1}(B'-b'_1)\right)\\
\le& 101M + 10H(A', B')\\
\le& 111M + 10H(A, B)\\
\le& 200H(A, B).
\end{aligned}$$

In the second inequality we use Proposition 8. In the third inequality we use the result of Lemma 6. In the second to last inequality we use $H(A', B') \le M + H(A, B)$, which follows from the definition of A' and B'. In the last inequality we use $H(A, B) \ge M$, which follows from the definition of M. This shows the lower bound in the statement of the lemma. We prove the upper bound now.

We have

$$\begin{aligned}
H(A, B) &\le M + H(A', B')\\
&\le M + 1000s^2 \max_i H\left(g^i_{a'_s-a'_1}(A'-a'_1), g^i_{b'_s-b'_1}(B'-b'_1)\right)\\
&\le M + 1000s^2 \left(\max_i H\left(g^i_{a_s-a_1}(A-a_1), g^i_{b_s-b_1}(B-b_1)\right) + 100M\right)\\
&\le 10^6 s^2 \cdot \max\left(M, H\left(g^i_{a_s-a_1}(A-a_1), g^i_{b_s-b_1}(B-b_1)\right)\right).
\end{aligned}$$

The second inequality follows from Lemma 6. The third inequality follows from Proposition 8. We established the upper bound of the lemma and the proof of Lemma 7 is complete. ◀

▶ **Theorem 9.** *There exists an embedding of $H_{s,1}$ into $\ell_\infty^{s^{O(s)}}$ with distortion $s^{O(s)}$.*

Proof. We will construct embedding f of H_s into $\ell_\infty^{s^{O(s)}}$ with distortion $s^{O(s)}$. Let $A = \{a_1, \dots, a_s\} \subseteq \mathbb{R}$ and $B = \{b_1, \dots, b_s\} \subseteq \mathbb{R}$ with $a_1 \le \dots \le a_s$ and $b_1 \le \dots \le b_s$. By Lemma 7, we can bound $H(A, B)$ in terms of the maximum of $|a_1 - b_1|$, $|a_s - b_s|$ and

$$H_{s-1}\left(g^i_{a_s-a_1}(A - a_1), g^i_{b_s-b_1}(B - b_1)\right)$$

over all $i \in [10s^2]$. By Lemma 7, we lose a factor $O(s^2)$ in the distortion. Notice that pointsets $g^i_{a_s-a_1}(A - a_1)$ and $g^i_{b_s-b_1}(B - b_1)$ are of size $s - 1$. That is, we decreased the number of points in the sets by 1. Also notice that the functions $g^i_{a_s-a_1}$ and $g^i_{b_s-b_1}$ depend only on sets A and B, respectively. The idea now is to apply this expression recursively until we arrive at pointsets of size 1, which we can embed into l_∞ trivially. More precisely, we define the following recursive embedding h^s of pointset A of size s. If $s \ge 2$,

$$h^s(A) := \left(a_1, a_s, h^{s-1}(g^1_{a_s-a_1}(A - a_1)), \dots, h^{s-1}(g^{10s^2}_{a_s-a_1}(A - a_1))\right).$$

If $s = 1$, then $h^1(A) = (a_1)$. $h^s(A)$ is concatenation of values a_1, a_s and $10s^2$ vector defined recursively by h^{s-1}. We define $f(A) := h^{|A|}(A)$. We call the recursive embedding at most s times, each time increases number of dimensions by a factor of $O(s^2)$ and the distortion by a factor of $O(s^2)$. This means that the final distortion is $\le [O(s^2)]^s \le s^{O(s)}$ and the dimension is $\le [O(s^2)]^s \le s^{O(s)}$. ◀

4 Embedding for Hausdorff metric over pointsets in $\mathbb{R}^d$

▶ **Theorem 10.** *There exists an embedding of $H_{s,d}$ into $\ell_\infty^{s^{O(s+d)}}$ with distortion $s^{O(s+d)}$ for an arbitrary integer $d \ge 1$.*

Proof. It suffices to consider the case $d > 1$, since the case $d = 1$ has been handled in the previous Section. Given sets $A, B \subseteq \ell_2^d$ of size $|A| = |B| = s$, we show how to produce sets $A_1, \dots, A_{2s^2+1}$ and $B_1, \dots, B_{2s^2+1}$ with the following properties.

1. Each A_i depends on A only. Each B_i depends on B only.
2. For every i, $A_i, B_i \subseteq \ell_2^{d-1}$ and $|A_i| = |B_i| = s$.
3. For every i, $H_{s,d-1}(A_i, B_i) \le H_{s,d}(A, B)$.
4. There exists i such that $H_{s,d-1}(A_i, B_i) \ge \frac{1}{Cs^2} H_{s,d}(A, B)$ for sufficiently large constant C.

From the properties we see that

$$H_{s,d}(A, B) \ge \max_{i=1,\dots,2s^2+1} H_{s,d-1}(A_i, B_i) \ge \frac{1}{Cs^2} H_{s,d}(A, B)$$

where A and B are any two subsets of d dimensional space and A_i, B_i are subsets of $d - 1$ dimensional space. If we repeat the construction $d - 1$ times in total, we get embedding that satisfies inequality

$$H_{s,d}(A, B) \ge \max_{j=1,\dots,(2s^2+1)^{d-1}} H_{s,1}(A'_j, B'_j) \ge \frac{1}{(Cs^2)^{d-1}} H_{s,d}(A, B).$$

Now we apply Theorem 9 to embed sets A'_j, B'_j into $l_\infty^{s^{O(s)}}$ with distortion $s^{O(s)}$. The final dimension of the embedding is $(2s^2+1)^{d-1} \cdot s^{O(s)} = s^{O(s+d)}$ as promised. The final distortion of the embedding is $(Cs^2)^{d-1} \cdot s^{O(s)} = s^{O(s+d)}$ as promised.

In the remainder of the proof we show how to construct the embedding with the four properties stated at the beginning of the proof. Consider the first two vectors of the standard basis of ℓ_2^d. These two vectors span a plane. Choose $2s^2+1$ unit vectors $v_1, \dots, v_{2s^2+1}$ in this plane so that angle between vectors v_i, v_{i+1} is $2\pi/(2s^2+1)$ for all $i = 1, \dots, 2s^2+1$ and v_1 is the first standard basis vector of ℓ_2^d. We define $v_{2s^2+2} := v_1$. We build A_i (B_i, resp.) by projecting A (B, resp.) on the hyperplane perpendicular to v_i for all $i = 1, \dots 2s^2+1$. The first property follows from the definition of A_i and B_i. The second property follows because every hyperplane of ℓ_2^d span ℓ_2^{d-1}. The third property follows because projection on hyperplane can only decrease interpoint distances. It remains to show the fourth property. Consider any pair of points $a \in A$ and $b \in B$. There can be at most two values i such that

$$\|\Pi_i(a) - \Pi_i(b)\|_2 < \frac{1}{10000s^2}\|a-b\|_2, \tag{17}$$

where Π_i denotes projection on the hyperplane defined by vector v_i. This is true because of the following considerations. Consider i such that inequality (17) does not hold. Then we must have

$$|v_i \cdot (a-b)| > \left(1 - \frac{1}{10000s^2}\right) \cdot \|v_i\|_2 \cdot \|a-b\|_2.$$

However, this can happen to at most two vectors v_i by the construction of v_i. Because there are $2s^2+1$ vectors v_i and at most s^2 pairs (a, b), $a \in A$, $b \in B$ determine distance $H_s(A, B)$, the fourth property follows. ◀

5 Probabilistic embedding

▶ **Theorem 11.** *For any $\alpha \in (0, 1/2)$ and integer $\Delta > 0$, there exists a probabilistic embedding f of H_s over subsets of $[\Delta]$ into 8Δ-dimensional ℓ_1 space $\ell_1^{8\Delta}$ that satisfies the following properties. For any two pointsets $A, B \subseteq [\Delta]$ with $|A| = |B| = s$,*

1. $\frac{1}{10}H_s^{1-\alpha}(A,B) \le \|f(A) - f(B)\|_1$*;*
2. $E[\|f(A) - f(B)\|_1] \le 100s/\alpha \cdot H_s^{1-\alpha}(A,B)$*,*

where the expectation in the second property is over the randomness of the embedding.

Proof. W.l.o.g. we assume that $\log_2 \Delta$ is positive integer. If this is not so, we increase Δ to $2^{\lceil \log_2 \Delta \rceil}$. For integer y and finite set $X \subseteq \mathbb{R}$, we define

$$y + X := \{x + y \mid x \in X\}.$$

For integer i, $0 \le i \le \log_2(2\Delta)$, and finite set $X \subseteq \mathbb{R}$, we define vector $f_i(X) \in \mathbb{R}^{\frac{2\Delta}{2^i}}$ as follows. For $l = 1, \dots, \frac{2\Delta}{2^i}$,

$$(f_i(X))_l = \begin{cases} 1 & \exists x \in X \text{ such that } (l-1)2^i < x \le l \cdot 2^i; \\ 0 & \text{otherwise.} \end{cases}$$

For integer v and finite set $X \subseteq \mathbb{R}$, we define embedding $f_v(X)$:

$$f_v(X) := \bigoplus_{i=0}^{\log_2(2\Delta)} 2^{i(1-\alpha)} f_i(v + X).$$

Embedding f is defined by choosing $v \in \{1, \dots, \Delta\}$ uniformly at random and setting $f(X) = f_v(X)$. Now we will show that the embedding satisfy the stated properties. Let $h = H_s(A, B)$.

The following lemma establishes the first inequality.

▶ **Lemma 12.** *For every v,*

$$\|f_v(A) - f_v(B)\|_1 \geq \frac{1}{10}h^{1-\alpha}.$$

Proof. We assume that $h \geq 2$. If $h = 1$, then $\|f_v(A) - f_v(B)\|_1 \geq 1 \geq \frac{1}{10}$. If $h = 2$, there is nothing to prove. W.l.o.g., let $a \in A$ be such that $d(a, B) = h$. Let $i = (\log_2 h) - 1 \geq 0$ and $l = \lceil \frac{a+v}{2^i} \rceil$. Because $a \in A$ and by the choice of l, we have

$$(f_i(v + A))_l = 1.$$

Because $d(a, B) = h$ and by the choice of i and l, we have

$$(f_i(v + B))_l = 0.$$

Therefore, we conclude:

$$\begin{aligned}\|f_v(A) - f_v(B)\|_1 &\geq 2^{i(1-\alpha)}\|f_i(v + A) - f_i(v + B)\|_1 \\ &\geq \left(\frac{h}{2}\right)^{1-\alpha} |\,(f_i(v + A))_l - (f_i(v + B))_l\,| \geq \frac{1}{10}h^{1-\alpha}.\end{aligned}$$ ◀

The following lemma establishes the second inequality.

▶ **Lemma 13.** $E[\|f(A) - f(B)\|_1] \leq 100s/\alpha \cdot h^{1-\alpha}$.

Proof.

▶ **Proposition 14.** *For every $i \in \{0, \ldots, \log_2(2\Delta)\}$,*

$$E_v[\|f_i(v + A) - f_i(v + B)\|_1] \leq 2s \min(1, 2h/2^i).$$

Proof. We define an undirected bipartite graph $G = (A, B, E)$ as follows. For every $a \in A$, we add edge (a, b), $b \in B$ such that $d(a, B) = |a - b|$. If there are multiple possibilities for b, we choose one b arbitrarily. For every $b \in B$, we add edge (a, b), $a \in A$ such that $d(b, A) = |a - b|$. If there are multiple possibilities for a, we choose one a arbitrarily.

By the definition of Hausdorff distance and f_i, we get

$$E_v[\|f_i(v + A) - f_i(v + B)\|_1] \leq \sum_{(a,b)\in E} \Pr_{v\in[\Delta]}\left[\left\lceil \frac{a+v}{2^i} \right\rceil \neq \left\lceil \frac{b+v}{2^i} \right\rceil\right].$$

We can upper bound every probability in the latter quantity by $\min(1, 2h/2^i)$ because for every $(a, b) \in E$, $|a - b| \leq h$. We get the bound stated in the proposition because $|E| \leq 2s$ by the definition of graph G. ◀

Using this proposition, we get

$$\begin{aligned}E[\|f(A) - f(B)\|_1] &\leq \sum_{i=0}^{\log_2(2\Delta)} 2^{i(1-\alpha)}E_v[\|f_i(v + A) - f_i(v + B)\|_1] \\ &\leq 2s \sum_{i=0}^{\log_2(2\Delta)} 2^{i(1-\alpha)} \min(1, 2h/2^i) \\ &\leq 2s \sum_{i=1}^{1+\log_2 h} \left(2^{i(1-\alpha)}\right) + 4sh \sum_{i=2+\log_2 h}^{\log_2(2\Delta)} 2^{-i\alpha} \\ &\leq 20sh^{1-\alpha} + 4sh \left(2^{-\alpha}\right)^{2+\log_2 h} \sum_{i=0}^{\infty} \left(2^{-\alpha}\right)^i\end{aligned}$$

$$\begin{aligned}&\leq 20sh^{1-\alpha} + \frac{20sh^{1-\alpha}}{\alpha}\\ &\leq 100s/\alpha \cdot h^{1-\alpha},\end{aligned}$$

which is what we needed. ◀

◀

▶ **Lemma 15.** *Let $U := \log_2 s + \log_2 h + 10$. Then*

$$\Pr_v[\forall i \geq U\ ,\ f_i(v+A) = f_i(v+B)] \geq 0.9.$$

Proof. Since $f_U(v+A) = f_U(v+B)$ implies that $f_i(v+A) = f_i(v+B)$ for all $i \geq U$, it is sufficient to prove that $\Pr_v[f_U(v+A) = f_U(v+B)] \geq 0.9$.

Let $G = (A, B, E)$ be the bipartite graph defined in Proposition 14. Since for all $(a,b) \in E$, $|a-b| \leq h$, we have

$$\begin{aligned}\Pr_v[f_U(v+A) = f_U(v+B)] &\geq \Pr_v\left[\forall (a,b) \in E\ ,\ \left\lceil \frac{a+v}{2^i} \right\rceil = \left\lceil \frac{b+v}{2^i} \right\rceil\right]\\ &\geq 1 - \sum_{(a,b)\in E} \Pr_v\left[\left\lceil \frac{a+v}{2^i} \right\rceil \neq \left\lceil \frac{b+v}{2^i} \right\rceil\right]\\ &\geq 1 - |E| \cdot \frac{2h}{2^U} \geq 1 - 2s \cdot \frac{2h}{2^U}\\ &\geq 0.9.\end{aligned}$$

◀

▶ **Lemma 16.** *Let $L := \log_2 h - \frac{1}{1-\alpha}\log s - 20$. For every $v \in \{1, \dots, \Delta\}$,*

$$\left\| \bigoplus_{i=0}^{L} 2^{i(1-\alpha)}(f_i(v+A) - f_i(v+B)) \right\|_1 \leq \frac{h^{1-\alpha}}{1000}.$$

Proof. We use the definition of $\oplus$ and L:

$$\sum_{i=0}^{L} \left\| 2^{i(1-\alpha)}(f_i(v+A) - f_i(v+B)) \right\| \leq 2s \cdot 5 \cdot 2^{L(1-\alpha)} \leq \frac{h^{1-\alpha}}{1000}.$$

◀

From Lemmas 15 and 16, we get that, with probability ≥ 0.9 the following happens. Almost all ℓ_1 mass of $\|f(A) - f(B)\|_1$ comes from $U - L - 1 \leq 100 \log_2 s$ vectors $f_i(v+A) - f_i(v+B)$. $f_i(v+A) - f_i(v+B)$ that correspond to $i \geq U$ or $i < L$ contribute at most $h^{1-\alpha}/1000$ to the ℓ_1 mass. Also notice that, by Theorem 11, the ℓ_1 mass of $f(A) - f(B)$ is at least $h^{1-\alpha}/10$. We get that we lose relatively small amount of ℓ_1 mass by discarding of many vectors f^i. We will use these observations in Theorem 23 below to reduce the dimensionality of the target ℓ_1 space in Theorem 11.

▶ **Definition 17.** Let $G = (A, B, E)$ be a bipartite graph. We call it r-regular if, for every vertex $a \in A$, the degree of a is equal to r.

▶ **Definition 18.** Graph $G = (A, B, E)$ is called random r-regular bipartite graph if it comes from a distribution defined by the following process. Initially, $E = \emptyset$. For every $a \in A$ we choose a subset of r distinct vertices of B uniformly at random and connect a to the all chosen vertices.

▶ **Definition 19.** Let $G = ([n], [m], E)$ be r-regular bipartite graph for some integers $r, n, m \geq 1$. We define matrix $\Phi_G \in \mathbb{R}^{m \times n}$ as follows.

$$(\Phi_G)_{i,j} = \begin{cases} \frac{1}{r} & \text{if } (i,j) \in E; \\ 0 & \text{otherwise.} \end{cases}$$

for all $i \in [m]$ and $j \in [n]$.

The following lemma can be shown using the probabilistic method.

▶ **Lemma 20.** *Let $G = ([n], [O(n/\delta^2)], E)$ be random r-regular bipartite graph for $r = O(1/\delta)$. For any subset X of vertices, let $N(X)$ denote the set of neighbors of vertices in X. Then we have*

$$\Pr_G[\forall X \subseteq A \ : \ |N(X)| \geq (1-\delta)r|X|] \geq 0.99.$$

The following result was shown in [3].

▶ **Theorem 21.** *Let $G = ([n], [m], E)$ be some r-regular bipartite graph with the property that*

$$\forall X \subseteq A \ : \ |N(X)| \geq (1-\delta)r|X|.$$

Let Φ_G be the matrix according as in Definition 19. Then we have that for all $x \in \mathbb{R}^n$,

$$(1 - O(\delta))\|x\|_1 \leq \|\Phi_G x\|_1 \leq \|x\|_1.$$

Below we will need the following lemma.

▶ **Lemma 22.** *Let $G = ([n'], O(n/\delta^2), E)$ be random r-regular bipartite graph for $r = O(1/\delta)$. Then for every $x \in \mathbb{R}^{n'}$ with $\|x\|_0 \leq n$ (number of non-zero entries of x is at most n),*

$$\Pr_G[(1 - O(\delta))\|x\|_1 \leq \|\Phi_G x\|_1 \leq \|x\|_1] \geq 0.99.$$

Proof. Consider matrix Φ_G restricted to the columns corresponding to the non-zero entries of x. This matrix correspond to random r-regular bipartite graph with at most n vertices on the left side. By Lemma 20, this matrix will satisfy the requirement for Theorem 21 with probability at least 0.99, concluding the proof. ◀

▶ **Theorem 23.** *For any $\alpha \in (0, 1/2)$ and integer $\Delta > 0$, there exists a probabilistic embedding f' of H_s into $\ell_1^{O(s \log s)}$ that satisfies the following properties. For any two pointsets $A, B \subseteq [\Delta]$ with $|A| = |B| = s$,*

1. $\frac{1}{100} H_s^{1-\alpha}(A,B) \leq \|f'(A) - f'(B)\|_1$ *with probability* $\geq 2/3$*;*
2. $E[\|f'(A) - f'(B)\|_1] \leq 100s/\alpha \cdot H_s^{1-\alpha}(A,B)$*.*

Proof. Let $C_1, C_2 > 0$ be large constants and $\delta > 0$ be a small enough constant that we will set later.

Let $G = ([8\Delta], \frac{C_1 \cdot 200 s \log_2 s}{\delta^2}, E)$ be random $\frac{C_2}{\delta}$-regular bipartite graph. By Lemma 22, for all $x \in \mathbb{R}^{8\Delta}$ with $\|x\|_0 \leq 200 s \log_2 s$,

$$\Pr_G[0.9 \cdot \|x\|_1 \leq \|\Phi_G x\|_1 \leq \|x\|_1] \geq 0.99, \tag{18}$$

where we choose C_1 and C_2 be large enough constants and $\delta > 0$ to be a small enough constant so that $1 - O(\delta) \geq 0.9$.

Let $f_v(X)$ be the embedding as in Theorem 11. For graph G and integer v, we define embedding $f'_{G,v}(X) := \Phi_G \cdot f_v(X)$. Embedding $f'(X)$ is defined by choosing uniformly random $v \in [\Delta]$ and G. Property 2 in the theorem follows since, for some G and v,

$$\|f'(A) - f'(B)\|_1 = \|\Phi_G \cdot (f_v(A) - f_v(B))\|_1 \leq \|f_v(A) - f_v(B)\|_1$$

and property 2 of Theorem 11. We used the fact that matrix Φ_G is a left stochastic matrix (that is, all columns of it sum up to 1) to conclude the inequality.

The remainder of the proof is devoted to show the first property. Consider entries of $f_v(A) - f_v(B)$ corresponding to embedding f_i for $i = U, \ldots, \log_2(2\Delta)$. By Lemma 15, with probability ≥ 0.9, all these entries are 0. We assume that this happens from now on. Consider entries of $f_v(A) - f_v(B)$ that correspond to embeddings f_i for $i = 0, \ldots, L$. By Lemma 16, the total sum of absolute values of these entries is upper bounded by $h^{1-\alpha}/1000$. We set all these entries (corresponding to $f_0, \ldots, f_L$) to 0. Because Φ_G is left stochastic, we change the value of $\|f'(A) - f'(B)\|_1$ by at most $h^{1-\alpha}/1000$. Now the only entries that are nonzero in $f_v(A) - f_v(B)$ correspond to $f_{L+1}, \ldots, f_{U-1}$. The total number of nonzero entries is at most

$$(|A| + |B|) \cdot (U - L - 1) \leq 2s \cdot 100 \log_2 s \leq 200 s \log_2 s.$$

By (18), we have that with probability ≥ 0.99,

$$\begin{aligned}\|f'(A) - f'(B)\|_1 &\geq 0.9 \cdot \|f_v(A) - f_v(B)\|_1 - h^{1-\alpha}/1000 \\ &\geq 0.9 \cdot \left(\frac{1}{10} h^{1-\alpha} - h^{1-\alpha}/1000\right) - h^{1-\alpha}/1000 \\ &\geq \frac{1}{100} h^{1-\alpha},\end{aligned}$$

which is what we need. In the second inequality we used the first property from Theorem 11 and the fact that we did set all entries, corresponding to f_i with $i \leq L$, to 0. The lower bound holds with probability $1 - 0.1 - 0.01 \geq 2/3$ by using the union bound. ◀

Acknowledgments. We thank Piotr Indyk for many helpful discussions.

References

1 Artūrs Bačkurs and Piotr Indyk. Better embeddings for planar earth-mover distance over sparse sets. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 280:280–280:289, New York, NY, USA, 2014. ACM. doi:10.1145/2582112.2582120.

2 Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

3 Radu Berinde, Anna C. Gilbert, Piotr Indyk, H. Karloff, and Martin J. Strauss. Combining geometry and combinatorics: A unified approach to sparse signal recovery. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 798–805. IEEE, 2008.

4 J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985. doi:10.1007/BF02776078.

5 Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. Pic: Practical internet coordinates for distance estimation. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 178–187. IEEE, 2004.

6 Aryeh Dvoretzky. *Some results on convex bodies and Banach spaces.* Hebrew University, 1960.

7 Martin Farach-Colton and Piotr Indyk. Approximate nearest neighbor algorithms for hausdorff metrics via embeddings. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 171–179. IEEE, 1999.

8 Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

9 P. Indyk and N. Thaper. Fast color image retrieval via embeddings. Workshop on Statistical and Computational Theories of Vision (at ICCV), 2003.

10 Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *focs*, page 10. IEEE, 2001.

11 Piotr Indyk. *High-dimensional Computational Geometry*. PhD thesis, Stanford University, 2001.

12 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

13 Piotr Indyk, Avner Magen, Anastasios Sidiropoulos, and Anastasios Zouzias. On-line embeddings. In *Proc. of APPROX*, 2010.

14 William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

15 Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

16 Ofer Neiman. Low dimensional embeddings of doubling metrics. In Christos Kaklamanis and Kirk Pruhs, editors, *Approximation and Online Algorithms*, volume 8447 of *Lecture Notes in Computer Science*, pages 12–23. Springer International Publishing, 2014.

17 Santosh S Vempala. *The random projection method*, volume 65. American Mathematical Soc., 2005.

Computing Approximate PSD Factorizations

Amitabh Basu*[1], Michael Dinitz†[2], and Xin Li‡[3]

1 **Dept. of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, USA**
`abasu9@jhu.edu`
2 **Dept. of Computer Science, Johns Hopkins University, Baltimore, USA**
`mdinitz@cs.jhu.edu`
3 **Dept. of Computer Science, Johns Hopkins University, Baltimore, USA**
`lixints@cs.jhu.edu`

Abstract

We give an algorithm for computing approximate PSD factorizations of nonnegative matrices. The running time of the algorithm is polynomial in the dimensions of the input matrix, but exponential in the PSD rank and the approximation error. The main ingredient is an *exact* factorization algorithm when the rows and columns of the factors are constrained to lie in a general polyhedron. This strictly generalizes nonnegative matrix factorizations which can be captured by letting this polyhedron to be the nonnegative orthant.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems, G.1.3 Numerical Linear Algebra

Keywords and phrases PSD rank, PSD factorizations

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.2

1 Introduction

Matrix factorization is a fundamental operation that has importance for diverse areas of mathematics and engineering such as machine learning, communication complexity, polyhedral combinatorics, statistical inference, and probability theory, to name a few. The problem can be stated quite simply as follows:

> Given two sequences of sets $\mathcal{K} = \{K_d\}_{d\in\mathbb{N}}$ and $\mathcal{K}' = \{K'_d\}_{d\in\mathbb{N}}$ where K_d, K'_d are subsets of $\mathbb{R}^d$ for all $d \in \mathbb{N}$, and a matrix $M \in \mathbb{R}^{n\times m}$, find a factorization $M = UV$ where $U \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{d\times m}$, and each row of U is in K_d and each column of V is in K'_d.

Such a factorization is called a $\mathcal{K}, \mathcal{K}'$ *factorization*. The smallest $d \in \mathbb{N}$ such that such a factorization exists is called the $\mathcal{K}, \mathcal{K}'$ *rank*. Most of the literature on this problem focuses on the case when the matrix M is nonnegative. In this context, when $K_d = K'_d = \mathbb{R}^d_+$, the factorization is called *nonnegative factorization*, and the corresponding rank is called *nonnegative rank*. When $K_d = K'_d$ are the cone of $d \times d$ PSD matrices, the factorization is known as a *PSD factorization* and the corresponding rank is called the *PSD rank*. These notions will be the object of study in this paper. A more general notion is that of *cone*

* Supported in part by the NSF grant CMMI1452820.
† Supported in part by NSF awards CCF-1464239 and CCF-1535887.
‡ Supported in part by NSF award CCF-1617713.

factorizations, where $\mathcal{K}$ is a family of cones and $\mathcal{K}'$ is the family of corresponding dual cones; see [11].

One of the most elegant applications of such factorizations arises in combinatorial optimization. A very common technique in approaching combinatorial optimization problems is to formulate the problem as a linear programming problem. However, a naive formulation of a problem may result in a polytope (the feasible region of the LP) with a large number of facets (exponentially many in the size of the problem), making it intractable to actually solve. One way around this is to try to express the polytope as the projection of a higher dimensional convex set. In particular, suppose that it can be expressed as the projection of either a higher dimensional polytope (LP), the feasible region of an SDP, or the feasible region of a more general convex optimization problem. Furthermore suppose that the number of "extra" dimensions is polynomial in the size of the original problem, and the description of the higher-dimensional convex optimization problem is also polynomial in the size of the original problem (i.e. there are not an exponential number of facets). Then we can efficiently solve the higher-dimensional problem, which means we can efficiently solve the original LP, even if its size makes solving it directly intractable.

It turns out that the smallest size of such a reformulation is a direct function of the nonnegative rank (for LP reformulations), the PSD rank (for SDP reformulations), or more general cone factorization ranks of the so-called *slack matrix* of the original LP formulation. The actual factorization can be used to explicitly find the smallest reformulation. This line of research started with a seminal paper by Yannakakis [24], and has recently seen a flurry of research activity – see the surveys [15, 7] and [10, 9, 21, 6, 16, 22, 17, 5] for some of the most recent breakthroughs.

In machine learning applications the actual factorization is perhaps more important than the value of the rank, as this factorization is key to certain text mining, clustering, imaging and bioinformatics applications. A key algorithmic question is computing such a factorization. Unfortunately, this question is computationally challenging – even computing the nonnegative rank was proved to be NP-hard by Vavasis [23].

A recent algorithmic breakthrough was achieved by Arora et al [1], where they showed that computing nonnegative factorizations can be done in polynomial time (in the dimensions of the input matrix) for the family of matrices with fixed (constant) nonnegative rank. The running time of their algorithm was doubly exponential in the nonnegative rank, and this was later improved to a singly exponential algorithm by Moitra [18], which he showed to be nearly optimal under the Exponential Time Hypothesis. The analogous question for PSD factorizations is largely open (the question is also posed in the survey [8]):

► **Question 1.** *Let $r \in \mathbb{N}$ be a constant. Does there exist an algorithm which, given any $n \times m$ nonnegative matrix M with PSD rank r, computes a PSD factorization of rank r in time polynomial in n, m?*

Our main result is a polynomial time algorithm to compute *approximate* factorizations of matrices with fixed PSD rank. We consider the space $\mathcal{S}^r$ of $r \times r$ symmetric matrices, and the cone of $r \times r$ PSD matrices in this space, denoted by $\mathcal{S}^r_+$. Given any matrix $M \in \mathbb{R}^{n \times m}$, we use the notation $\|M\|_\infty := \max_{i,j} |M_{ij}|$.

More precisely, we prove:

► **Theorem 2.** *There exists an algorithm which, given any $\epsilon > 0$ and any $n \times m$ nonnegative matrix M with PSD rank r, computes a factorization $M = UV$ such that each row of U and each column of V are in $\mathcal{S}^r_+$ such that*

$$\|M - UV\|_\infty \leq \epsilon \|M\|_\infty$$

and has runtime $O((4mn)^{f_1(\epsilon,r)}(2mn+f_2(\epsilon,r))^{f_3(\epsilon,r)})+(m+n)(r^3+r^2\log^2 r\log(\frac{1}{\epsilon}))$*, where* $f_1(\epsilon,r)=r^2(r^a(\frac{1}{\epsilon})^{rb})^{r^2+2^{r^2}}$*,* $f_2(\epsilon,r)=4r^2(r^a(\frac{1}{\epsilon})^{rb})^{r^2+1}$*, and* $f_3(\epsilon,r)=2r^2(r^a(\frac{1}{\epsilon})^{rb})^{r^2}$ *for some universal constants* a *and* b*. Note that* $f_1(\epsilon,r), f_2(\epsilon,r)$ *and* $f_3(\epsilon,r)$ *are independent of* m *and* n*, so the runtime becomes polynomial in the dimensions of* M *if* r,ϵ *are fixed constants.*

Approximate PSD factorizations can be useful for reformulation questions in combinatorial optimization, where one seeks approximations of the original polyhedron using SDPs, as opposed to an exact reformulation – see [12] for results along this direction. In particular, approximate factorizations of the slack matrix of a polytope can sometimes be used to compute “inner" and “outer" approximations of the polytope, each of which can then be optimized over in order to give an approximation to the true optimal solution of the polytope. However, in [12], these approximations are guaranteed only when the corresponding matrix factorization error is calculated in certain induced matrix norms (in particular the $\|\cdot\|_{1,2}$ and $\|\cdot\|_{1,\infty}$ norms). Consequently, it is unclear if the approximate factorizations generated by Theorem 2 give similar results, primarily since our notion of an approximate factorization involves the $\|\cdot\|_\infty$-norm rather than the appropriate induced matrix norms. Thus while our results do not directly imply any new approximation algorithms, they do provide ideas on how to go beyond nonnegative factorizations to PSD or more general conic factorizations, if one admits approximate factorizations as opposed to exact ones.

1.1 Technical overview

Our algorithm for Theorem 2 is inspired by ideas behind the algorithm in Arora et al [1]. However, there are some important differences. Arora et al's algorithm uses properties of the nonnegative orthant that do not hold for the cone of PSD matrices. To overcome this difficulty, we need to approximate the PSD cone by a polyhedral cone obtained by intersecting enough tangent halfspaces. We then generalize Arora et al's techniques to compute factorizations inside a general polyhedron, as opposed to just the nonnegative orthant. The nonnegative orthant is a very special polyhedron, and many of its special properties are utilized in the algorithm of Arora et al. We have to use interesting techniques from polyhedral theory (such as Fourier-Motzkin elimination) to extend these ideas to handle general polyhedra (see Theorem 14). Finally, to bound the errors in the approximate factorization, we use some technical results on rescaling PSD factorizations due to Briet et al [6] (Theorem 17).

1.2 Model of computation

We will present our algorithm from Theorem 2 in the real arithmetic model of computation developed by Blum, Shub and Smale [4], thus ignoring questions of approximating irrational computations by rational arithmetic. This is just for the ease of exposition. In Section 4.1, we show that Theorem 2 can be proved by designing an algorithm that operates in the more standard Turing machine model of computation.

2 Preliminaries

For any normed space $(V,\|\cdot\|)$, we denote the distance between two subsets $X,Y\subseteq V$ by $dist(X,Y):=\inf\{\|x-y\|: x\in X, y\in Y\}$. A closed subset P of a normed space V is called a *closed cone* if it is convex and $\lambda P\subseteq P$ for all $\lambda\geq 0$. A cone is called a *polyhedral cone* if it is the intersection of finitely many halfspaces. For any closed cone P in an inner product space $(V, \langle \cdot, \cdot \rangle)$ (closed with respect to the norm obtained from the inner product), the dual cone will be denoted by

$$P^* = \{v \in V : \langle v, y \rangle \geq 0 \ \ \forall y \in P\}.$$

We recall a standard fact about dual cones:

▶ **Fact 3.** *Let $(V, \langle \cdot, \cdot \rangle)$ be an inner product space with $\|\cdot\|$ denoting the norm on V induced by the inner product. For any closed cone $P \subseteq V$, if $x \in V$ such that $dist(x, P) = \delta$, then there exists a vector $a \in P^*$ with $\|a\| = 1$ such that the distance of x from the hyperplane $\{y \in V : \langle a, y \rangle = 0\}$ is δ, i.e., $\langle a, x \rangle = -\delta$.*

On the space $\mathcal{S}^r$ of $r \times r$ symmetric matrices, we consider the inner product $\langle A, B \rangle = \sum_{i,j} A_{ij} B_{ij}$.

▶ **Fact 4.** *The PSD cone $\mathcal{S}^r_+$ is self-dual, i.e., $(\mathcal{S}^r_+)^* = \mathcal{S}^r_+$.*

▶ **Definition 5.** Let C be a subset of a normed space $(V, \|\cdot\|)$. For $\epsilon > 0$, $\mathcal{X}_\epsilon \subseteq C$ is called an ϵ-covering for C with respect to the norm $\|\cdot\|$ if for every $a \in C$, there exists $a' \in \mathcal{X}_\epsilon$ such that $\|a - a'\| < \epsilon$.

▶ **Definition 6.** For any closed cones $P_1 \subseteq P_2$ in a normed space $(V, \|\cdot\|)$, we say P_2 is an ϵ-approximation of P_1 with respect to $\|\cdot\|$ for some $\epsilon > 0$, if for every $p_2 \in P_2$, there exists a point $p_1 \in P_1$ such that $\|p_2 - p_1\| \leq \epsilon \|p_2\|$.

▶ **Theorem 7.** *Let $C = \{x \in \mathcal{S}^r_+ : \|x\|_2 = 1\}$ be the spherical cap on the PSD cone. Let $\epsilon > 0$ and let $\mathcal{X}_\epsilon \subseteq C$ be any* finite *ϵ-covering for C with respect to some norm $\|\cdot\|$. Then the polyhedral cone*

$$P := \{x \in \mathcal{S}^r : \langle a', x \rangle \geq 0 \ \ \forall a' \in \mathcal{X}_\epsilon\}$$

is an ϵ-approximation for $\mathcal{S}^r_+$ with respect to $\|\cdot\|$.

Proof. It suffices to prove that for any $x \in \mathcal{S}^r$ such that $dist(x, \mathcal{S}^r_+) > \epsilon \|x\|$, then $x \notin P$. By Fact 3 and Fact 4, there exists $a \in C$ such that $\langle a, x \rangle < -\epsilon \|x\|$. By definition of ϵ-covering, there exists $a' \in \mathcal{X}_\epsilon$ such that $\|a - a'\| < \epsilon$. By Cauchy-Schwartz, we have $|\langle a', x \rangle - \langle a, x \rangle| \leq \|a - a'\| \|x\| < \epsilon \|x\|$. Combined with $\langle a, x \rangle < -\epsilon \|x\|$, this implies $\langle a', x \rangle < 0$. Thus, by definition of P, $x \notin P$. ◀

▶ Remark. Since C is a compact set, there always exists a finite ϵ-covering of C for any $\epsilon > 0$. Rabani and Shpilka [20] give explicit constructions of small ϵ-coverings of the sphere $\mathcal{S}^{d-1} = \{x \in \mathbb{R}^d : \|x\|_2 = 1\}$, which will prove useful for us. In particular, their results imply the following bound.

▶ **Theorem 8.** *There exists a polyhedral ϵ-approximation of $\mathcal{S}^r_+$ with respect to the $\|\cdot\|_\infty$ norm of size $O(r^a \epsilon^{-rb})$ for some universal constants a, b.*

The following fact from linear algebra will be used.

▶ **Proposition 9.** *Any linear transformation $T : \mathbb{R}^d \to \mathbb{R}^m$ can be expressed as $T = A \circ \phi$ where $\phi : \mathbb{R}^d \to \ker(T)^\perp$ is the projection of $\mathbb{R}^d$ onto $\ker(T)^\perp$ and $A : \ker(T)^\perp \to \mathrm{Im}(T)$ is an invertible linear transformation.*

This leads to the following observation about linear transformation of polyhedra.

▶ **Proposition 10.** *Let $P \subseteq \mathbb{R}^d$ be a polyhedron defined by p inequalities. Let $T : \mathbb{R}^d \to \mathbb{R}^m$ be any linear transformation. Then $T(P)$ is a polyhedron defined by at most $O(p^{2^d})$ inequalities.*

Proof. Let us make a change of coordinates such that $\ker(T)^\perp = \mathbb{R}^{d'}$ with $d \geq d' \geq 0$ - this does not change the number of inequalities required to describe P or $T(P)$. By Proposition 9, T can be expressed as $A \circ \phi$ where ϕ is the projection from $\mathbb{R}^d \to \mathbb{R}^{d'}$, and A is an invertible transformation from $\mathbb{R}^{d'} \to \mathrm{Im}(T)$. So we just need to analyze the effect of ϕ and A on the number of inequalities.

To analyze $\phi(P)$, we note that the Fourier-Motzkin elimination process [25] implies that projecting out a single variable can be done by squaring the number of inequalities. By repeatedly applying this, we get that $\phi(P)$ has at most $p^{2^{d-d'}}$ inequalities. Since A is an invertible linear transformation, $A(\phi(P))$ has the same number of inequalities as $\phi(P)$. The result follows. ◀

We list one final linear algebraic observation. Let $\dim(W)$ denote the dimension of an affine subspace W, and let $\mathrm{aff}(X)$ denote the affine hull of the columns of a matrix X (or just a finite set of vectors X).

▶ **Proposition 11.** *Let $\{m_1, \dots, m_t\} \subseteq \mathbb{R}^m$ and $\{b_1, \dots, b_t\} \subseteq \mathbb{R}^d$ such that there exists a linear transformation $A : \mathbb{R}^d \to \mathbb{R}^m$ satisfying $m_i = A(b_i)$ for all $i = 1, \dots, t$. Further suppose that $\dim(\mathrm{aff}(\{m_1, \dots, m_t\})) = \dim(\mathrm{aff}(\{b_1, \dots, b_t\})) = k$ and that $m_1, \dots, m_{k+1}$ and $b_1, \dots, b_{k+1}$ are maximal affinely independent subsets, respectively. Then, for every $i > k+1$, if m_i is expressed as an affine combination $m_i = \lambda_1 m_1 + \ldots + \lambda_{k+1} m_{k+1}$ with $\sum_{j=1}^{k+1} \lambda_j = 1$, then $b_i = \lambda_1 b_1 + \ldots + \lambda_{k+1} b_{k+1}$. In other words, b_i is an affine combination of $b_1, \dots, b_{k+1}$ with the same coefficients as in the expression for m_i.*

The following result about projecting onto the PSD cone will be used [14].

▶ **Proposition 12.** *Let C be an $r \times r$ symmetric matrix with spectral decomposition $U \Lambda U^T$, where U is the matrix with the eigenvectors of C as columns, and $\Lambda = \mathrm{Diag}(\lambda_1, \dots, \lambda_r)$ is the diagonal matrix with eigenvalues of C on the diagonals. Then, the matrix $C^* = U \left(\mathrm{Diag}(\max\{0, \lambda_1\}, \dots, \max\{0, \lambda_r\})\right) U^T$ is the closest matrix in $\mathcal{S}_+^r$ to C with respect to the $\|\cdot\|_2$ norm.*

We also use the following deep result from real algebraic geometry and quantifier elimination.

▶ **Theorem 13** ([2])**.** *There is an algorithm that tests the feasibility of any system of s polynomial equalities involving N variables with d as the maximum degree of any polynomial, that runs in time $(sd)^{O(N)}$.*

3 Factorizations from a polyhedron

Our main tool for proving Theorem 2 will be the following generalization of the algorithm of Arora et al. [1], who proved it for the special case of P being the nonnegative cone. We generalize this to an arbitrary polyhedron P.

▶ **Theorem 14.** *Let M be an $n \times m$ matrix with nonnegative entries, and let P be some polyhedron in $\mathbb{R}^d$ described by p inequalities. If there exists a factorization $M = UV$ such that each row of U and each column of V is in P, then one can compute such a factorization in time $O((4mn)^{d(p^{d+2^d})}(2mn + 4dp^{d+1})^{2p^d d^2})$, which is polynomial in m, n if d and p are fixed.*

In order to prove this theorem, we first need a few useful lemmas. Let X be a $p \times q$ matrix. For any subset $C \subseteq \{1, \ldots q\}$, let X^C denote the matrix formed by the subset of columns indexed by C. Similarly, for any subset $R \subseteq \{1, \ldots p\}$, let X_R denote the matrix formed by the rows indexed by R.

► **Lemma 15.** *Let M be an $n \times m$ matrix with nonnegative entries. Let P be some polyhedron in $\mathbb{R}^d$ described by p inequalities. Suppose there exists a factorization $M = UV$ such that each row of U and each column of V is in P. Then there exists a partition $C_1 \uplus C_2 \uplus \ldots \uplus C_k = \{1, \ldots, m\}$, a partition $R_1 \uplus R_2 \uplus \ldots \uplus R_\ell = \{1, \ldots, n\}$, and matrices $\bar{U} \in \mathbb{R}^{n \times d}$, $\bar{V} \in \mathbb{R}^{d \times m}$ such that the following properties all hold:*

1. *$M = \bar{U}\bar{V}$.*
2. *Each row of $\bar{U}$ and each column of $\bar{V}$ is in P. Moreover, there exist faces $F_1, \ldots, F_k$ of P such that for every $j = 1, \ldots, k$, the columns of $\bar{V}^{C_j}$ all lie on F_j. Similarly, there exist faces $G_1, \ldots, G_\ell$ such that for every $i = 1, \ldots, \ell$, the rows of $\bar{U}_{R_i}$ all lie on G_i.*
3. *$\dim(\mathrm{aff}(M^{C_j})) = \dim(\mathrm{aff}(\bar{V}^{C_j}))$ for all $j = 1, \ldots, k$.*
4. *$\dim(\mathrm{aff}((M_{R_i})^T)) = \dim(\mathrm{aff}((\bar{U}_{R_i})^T))$ for all $i = 1, \ldots, \ell$.*
5. *$k, \ell \leq p^d$.*

Proof. We use an idea from Arora et al. [1] to produce $\bar{U}, \bar{V}$ and the partitions with the stated properties. Starting from U, V, we will first construct $\bar{V}$, and then use this to construct $\bar{U}$. Slightly more formally, we will first construct a partition $C_1 \uplus C_2 \uplus \ldots \uplus C_k = \{1, \ldots, m\}$, a matrix $\bar{V}$ such that $M = U\bar{V}$, and exhibit faces $F_1, \ldots, F_k$ such that columns of $\bar{V}^{C_j}$ are contained in F_j for all j, and also condition 3 in the statement is satisfied. We will then construct a partition $R_1 \uplus R_2 \uplus \ldots \uplus R_\ell = \{1, \ldots, n\}$, a matrix $\bar{U}$ such that $M = \bar{U}\bar{V}$, and exhibit faces $G_1, \ldots, G_\ell$ such that rows of $\bar{U}^{R_i}$ are contained in G_i for all i, and condition 4 from the statement is satisfied. Condition 5 will then be verified.

For any $p \in P$, let F_p be the face of P of minimum dimension containing p. This induces a partial ordering $\succ$ on the points in P, where $p_1 \succ p_2$ if $F_{p_1} \supsetneq F_{p_2}$.

For every column v^j of V, consider the set $(v^j + \ker(U)) \cap P$ and define $\bar{v}^j$ to be a minimal element in this set according to this partial order. Note that for any $p \in P$, if there exists $u \in \ker(U) \setminus \{0\}$ such that the line $p + \lambda u$, $\lambda \in \mathbb{R}$ lies in the affine hull of F_p, then one can choose λ such that $p + \lambda u$ is in a strict face of F_p. Thus, by the minimal choice of $\bar{v}^j$, we have that $(\bar{v}^j + \ker(U)) \cap \mathrm{aff}(F_{\bar{v}^j}) = \bar{v}^j$ for every $j \in \{1, \ldots, m\}$. We set $\bar{V}$ to be the matrix with columns $\bar{v}^j$. Note that $M = U\bar{V}$ as desired, since $U\bar{v}^j = U(v^j + x^j) = Uv^j$ for every $j \in \{1, \ldots, m\}$, where x^j is some vector in $\ker(U)$.

The partition $C_1 \uplus C_2 \uplus \ldots \uplus C_k$ of the columns of $\bar{V}$ is obtained by grouping the columns together based on the face of minimum dimension that they lie on. Thus, $k \leq p^d$ which is an upper bound on the number of faces of P. Moreover, these faces of minimum dimension will form the faces $F_1, \ldots, F_k$ in Condition 2. We now need to verify that $\dim(\mathrm{aff}(M^{C_j})) = \dim(\mathrm{aff}(\bar{V}^{C_j}))$ for all $j = 1, \ldots, k$. Fix some j and let the columns of M^{C_j} be $\{m_0, m_1, \ldots, m_h\}$ and let the columns of $\bar{V}^{C_j}$ be $v_0, v_1, \ldots, v_h$. Let $\dim(\mathrm{aff}(\bar{V}^{C_j})) = k$, and without loss of generality assume that $v_0, \ldots, v_k$ are affinely independent. Since $M^{C_j} = U\bar{V}^{C_j}$, we know that $\dim(\mathrm{aff}(M^{C_j})) \leq \dim(\mathrm{aff}(\bar{V}^{C_j}))$. If the inequality is strict, then the columns $m_0, \ldots, m_k$ are affinely *dependent*, and thus there exist $\lambda_0, \ldots, \lambda_k \in \mathbb{R}$ not all zero such that and $\lambda_0 + \lambda_1 + \ldots + \lambda_k = 0$ and $\lambda_0 m_0 + \lambda_1 m_1 + \ldots, \lambda_k m_k = 0$. Since $M^{C_j} = U\bar{V}^{C_j}$, the vector $v = \lambda_0 v_0 + \lambda_1 v_1 + \ldots, \lambda_k v_k$ satisfies $v \in \ker(U) \setminus \{0\}$ ($v \neq 0$ because $v_1, \ldots, v_k$ are affinely independent). Recall that F_j is the face of minimum dimension containing $v_0, v_1, \ldots, v_h$, we find that $v_0 + \lambda v$, $\lambda \in \mathbb{R}$ lies in the affine hull of F_j. This would contradict the construction of the columns of $\bar{V}$. Therefore, $\dim(\mathrm{aff}(M^{C_j})) = \dim(\mathrm{aff}(\bar{V}^{C_j}))$.

In a similar manner, we can change the rows of U (keeping $\bar{V}$ fixed) to obtain $\bar{U}$ so that conditions 1 still holds, and there is a partition $R_1 \uplus R_2 \uplus \ldots \uplus R_\ell = \{1, \ldots, n\}$ and faces $G_1, \ldots, G_\ell$ so that Conditions 2 and 4 are satisfied. Finally, as was the case with the column partition, the upper bound on the total number of faces of P gives $\ell \leq p^d$. This completes the construction. ◀

Let X be a a set of points in $\mathbb{R}^d$. We say a set of polyhedra $P_1, \ldots, P_k$ is a *polyhedral covering of* X if $(P_1 \cap X) \cup \ldots \cup (P_k \cap X) = X$ – note that $P_1, \ldots, P_k$ do not have to be disjoint polyhedra. We say a partition $X_1 \uplus \ldots \uplus X_k = X$ is *induced by a polyhedral covering* if there exists a polyhedral covering $P_1, \ldots, P_k$ of X such that $X_1 = P_1 \cap X$ and $X_i = (P_i \cap X) \setminus (X_1 \cup \ldots \cup X_{i-1})$ for $i = 2, \ldots, k$. A (k_1, k_2)*-polyhedral partition of* X is a partition induced by a polyhedral covering of X with at most k_1 polyhedra and each polyhedron is described by at most k_2 inequalities.

▶ **Lemma 16.** *Let k_1, k_2 be fixed natural numbers and let X be a set of points in $\mathbb{R}^d$. The number of (k_1, k_2)-polyhedral partitions is at most $O((2^d m^d)^{k_1 k_2})$ and one can enumerate these partitions in time $O((2^d m^d)^{k_1 k_2})$, where $m = |X|$.*

Proof. Let us first count the number of subsets of X of the form $P \cap X$ where P is a polyhedron with at most k_2 inequalities. As observed in Arora et al [1], this can be reduced to counting the number of subsets of the form $H \cap X$ where H is a halfspace. The number of such subsets is $O(2^d m^d)$ and can be enumerated in the same amount of time (as was shown in Arora et al [1] by a simple iterative procedure). To choose a subset of the form $P \cap X$ where P is a polyhedron with at most k_2 inequalities, one simply needs to iteratively choose k_2 subsets given by halfspace intersections. Thus, there are $O((2^d m^d)^{k_2})$ such subsets and these can be enumerated in this iterative fashion.

To finally get partitions induced by polyhedral coverings, one needs to iteratively choose k_1 subsets of the form $P \cap X$ where P is a polyhedron with at most k_2 inequalities. The result follows. ◀

Using these tools, we can now prove Theorem 14.

Proof of Theorem 14. By Lemma 15, there exists a partition $C_1 \uplus C_2 \uplus \ldots \uplus C_k = \{1, \ldots, m\}$, a partition $R_1 \uplus R_2 \uplus \ldots \uplus R_\ell = \{1, \ldots, n\}$, and matrices $\bar{U} \in \mathbb{R}^{n \times d}$, $\bar{V} \in \mathbb{R}^{d \times m}$ such that conditions $1 - 5$ in Lemma 15 hold. Our algorithm will find these partitions, as well as the matrices $\bar{U}$ and $\bar{V}$. By conditions 1 and 2 of Lemma 15, these matrices form the desired factorization of M.

Let $F_1, \ldots, F_k, G_1, \ldots, G_\ell$ be the faces of P referred to in Condition 2 of Lemma 15. For any fixed $j \in \{1, \ldots, k\}$, since $M^{\{s\}} = \bar{U}\bar{V}^{\{s\}}$ for every $s \in C_j$, we have $\{M^{\{s\}} : s \in C_j\} \subseteq \bar{U}(F_j)$ by Condition 2 from Lemma 15. Invoking Proposition 10, we obtain that $\bar{U}(F_j)$ is described using at most p^{2^d} inequalities. By Lemma 15, k is bounded by p^d. Therefore, $M^{C_1}, \ldots, M^{C_k}$ is a (p^d, p^{2^d})-polyhedral partition of $\{M^1, \ldots, M^m\}$ using the polyhedra $\bar{U}(F_1), \ldots, \bar{U}(F_k)$. Similarly, $M_{R_1}, \ldots, M_{R_\ell}$ is a (p^d, p^{2^d})-polyhedral partition of $\{M_1, \ldots, M_n\}$ using the the polyhedra $\bar{V}^T(G_1), \ldots, \bar{V}^T(G_\ell)$.

Lemma 16 implies that we can enumerate all possible (p^d, p^{2^d})-polyhedral partitions of $\{M^1, \ldots, M^m\}$ in time $O((2^d m^d)^{p^{d+2^d}})$. Similarly, one can enumerate all possible partitions (p^d, p^{2^d})-polyhedral partitions of $\{M_1, \ldots, M_n\}$ in time $O((2^d n^d)^{p^{d+2^d}})$.

Condition 3 from Lemma 15 and Proposition 11 imply that for each $j \in \{1, \ldots, k\}$, there exist $\dim(\text{aff}(M^{C_j})) + 1 \leq d + 1$ columns of $\bar{V}^{C_j}$, such that every other column in $\bar{V}^{C_j}$ can be expressed as affine combinations of these columns. Moreover, the coefficients in these affine combinations can be computed from the columns of M^{C_j}. Similarly, Condition 4 from Lemma 15 and Proposition 11 imply that for every $i = 1, \ldots, \ell$, the rows of $\bar{U}_{R_i}$ can be expressed as known affine combinations of $\dim(\mathrm{aff}(M_{R_j})) + 1 \leq d + 1$ rows of $\bar{U}_{R_i}$.

We first make a guess for the partitions by enumerating all possible (p^d, p^{2^d})-polyhedral partitions $C_1, \ldots, C_k$ of the columns of M and all possible (p^d, p^{2^d})-polyhedral partitions $R_1, \ldots, R_\ell$ of the rows of M. For each choice of such partitions, introduce variables for the entries of the $\dim(\mathrm{aff}(M^{C_j})) + 1$ special columns of $\bar{V}^{C_j}$, $j = 1, \ldots, k$, and $\dim(\mathrm{aff}(M_{R_j})) + 1$ special rows of $\bar{U}_{R_i}$, $i = 1, \ldots, \ell$ which would correspond to the appropriate affinely independent columns in M^{C_j} and rows in M_{R_i} respectively. We now need to ensure that $M = \bar{U}\bar{V}$. For this we set up a system of mn quadratic constraints $M_{ij} = \langle \bar{U}_i, \bar{V}^j \rangle$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, m$, where the entries of $\bar{U}_i$ and $\bar{V}^j$ are expressed in terms of the variables the $\dim(\mathrm{aff}(M^{C_j})) + 1$ special columns of $\bar{V}^{C_j}$, and $\dim(\mathrm{aff}(M_{R_j})) + 1$ special rows of $\bar{U}_{R_i}$ as discussed above. Notice that this system has only $O((k + \ell)d^2)$ variables. We finally also impose the condition that these special columns are in P, which corresponds to $(k + \ell)dp$ more linear inequalities in these variables. We finally invoke Theorem 13 to test the feasibility of such a system. This process is then repeated for all possible (p^d, p^{2^d})-polyhedral partitions $C_1, \ldots, C_k$ of the columns of M and (p^d, p^{2^d})-polyhedral partitions $R_1, \ldots, R_\ell$ of the rows of M, until we find one that satisfies all the conditions in Lemma 15. ◀

4 Proof of Theorem 2

For any square matrix $X \in \mathbb{R}^{r \times r}$, we use $\|X\|_{sp} := \max_{y \in \mathbb{R}^r \setminus \{0\}} \frac{\|Xy\|_2}{\|y\|_2}$ to denote the spectral norm of X. The algorithm depends on this key result (paraphrased here) from [6].

▶ **Theorem 17.** *Let M be an $n \times m$ matrix with nonnegative entries. If M has a PSD factorization $M = UV$ such that the rows of U and columns of V are in $\mathcal{S}^r_+$, then there exists a PSD factorization $M = \bar{U}\bar{V}$ such that the rows of $\bar{U}$ and the columns of $\bar{V}$ have spectral norm bounded by $\sqrt{r\|M\|_\infty}$.*

We outline the steps of the algorithm in Theorem 2. Let $f(r)$ be such that for every matrix $X \in \mathcal{S}^r$, $\|X\|_\infty \leq f(r)\|X\|_{sp}$. Such an $f(r)$ must exist because all norms are equivalent on a Euclidean space, i.e., their values are the same upto a factor depending only on the dimension of the space.

1. Given M, let $\Delta = \|M\|_\infty$. Construct a polyhedral ϵ-approximation of $\mathcal{S}^r_+$ with respect to the $\|\cdot\|_\infty$ norm on $\mathcal{S}^r$ using Theorem 8. Let P be the polyhedron formed by the intersection of this polyhedral approximation with the cube $\{x \in \mathcal{S}^r : \|x\|_\infty \leq f(r)\sqrt{r\Delta}\}$.
2. By Theorem 17 and the assumption that M has PSD rank r, we know there exists a factorization $M = \bar{U}\bar{V}$ such that the rows of $\bar{U}$ and columns of $\bar{V}$ are in the PSD cone, and their spectral norm is at most $\sqrt{r\Delta}$. Therefore, for every row u of $\bar{U}$, we have $\|u\|_\infty \leq f(r)\sqrt{r\Delta}$ and similarly for the columns of $\bar{V}$. This implies that the rows of $\bar{U}$ and columns of $\bar{V}$ are in P. Since $\bar{U}, \bar{V}$ exist, we can employ Theorem 14 to construct a factorization $M = U'V'$ such that the rows of U' and the columns of V' are in P. Note that the algorithm of Theorem 14 may not produce a PSD factorization. To obtain an approximate PSD factorization, we construct matrices U and V by projecting each row of U' to the nearest point in the PSD cone (according to the $\|\cdot\|_\infty$ norm), and similarly for the columns of V'. This can be done in polynomial time by invoking Proposition 12.

This concludes the description of the algorithm. The running time claimed by Theorem 2 is tedious but straightforward to verify. It remains to prove that

$$\|M - UV\|_\infty \le \epsilon \|M\|_\infty.$$

Our first step will be to use the fact that we projected from an ϵ-approximation to the PSD cone. In particular, we know that $\|V'^j - V^j\|_\infty \le \epsilon\|V'^j\|_\infty$ for each $j \in \{1, \ldots, m\}$, and similarly $\|U'_i - U_i\|_\infty \le \epsilon\|U'^j\|_\infty$ for each $i \in \{1, \ldots, n\}$. This clearly implies that

$$\|V' - V\|_\infty \le \epsilon\|V'\|_\infty \text{ and } \|U' - U\|_\infty \le \epsilon\|U'\|_\infty. \tag{1}$$

Now we can analyze the approximation of our factorization:

$$\begin{aligned} \|M - UV\|_\infty &= \|U'V' - UV\|_\infty \\ &\le \|U'V' - U'V\|_\infty + \|U'V - UV\|_\infty \\ &\le r\|U'\|_\infty\|V' - V\|_\infty + r\|U' - U\|_\infty\|V\|_\infty \\ &\le r\|U'\|_\infty(\epsilon\|V'\|_\infty) + r\epsilon\|U'\|_\infty\|V\|_\infty \end{aligned} \tag{2}$$

where the first equality is from the fact that $M = U'V'$, the first inequality is from the triangle inequality, and the third is from (1). The second inequality follows from the observation that for any matrices $A \in \mathbb{R}^{n\times r}, B \in \mathbb{R}^{r\times m}$, $\|AB\|_\infty \le r\|A\|_\infty\|B\|_\infty$.

Since $\|V\|_\infty \le (1+\epsilon)\|V'\|_\infty$ because of (1), we obtain $\|M - UV\|_\infty \le 3\epsilon r\|U'\|_\infty\|V'\|_\infty$. Since each row u of U' is in P, we have $\|u\|_\infty \le f(r)\sqrt{r\Delta}$. Therefore, $\|U'\|_\infty \le f(r)\sqrt{r\Delta}$. Similarly, $\|V'\|_\infty \le f(r)\sqrt{r\Delta}$. Hence, $\|M - UV\|_\infty \le 3\epsilon r\|U'\|_\infty\|V'\|_\infty \le 3f(r)r^2\epsilon\Delta$. By redefining ϵ appropriately (in particular, letting ϵ' be the previous ϵ and letting $\epsilon = 3f(r)f\epsilon'$), we get that

$$\|M - UV\|_\infty \le \epsilon\|M\|_\infty$$

as desired.

4.1 Computing on a Turing Machine

As mentioned in the introduction, the algorithm described above works in the real arithmetic model of computation. However, this was only for ease of exposition. We now show how to remove this assumption and work in the more standard Turing machine model of computation.

The assumption of real arithmetic was used in two places. First, it was used when invoking Theorem 13 to solve a system of polynomial inequalities in the proof of Theorem 14. The second time it was used was for computing the spectral decompositions in Proposition 12 while projecting to the PSD cone in Step 2 above.

The first problem can be resolved by using a result of Grigor'ev and Vorobjov [13] which states that one can compute rational approximations to solutions of polynomial systems with integer coefficients within δ accuracy for any rational $\delta > 0$, in time that is polynomial in the parameters $\log(\frac{1}{\delta})$, maximum bit length of the coefficients, and $(sd)^{N^2}$, where s is the number of inequalities, d is the maximum degree, and N is the number of variables (See "Remark" at the end of page 2 in [13]). This implies that one can find rational approximations for the rows and columns of U' and V' in Step 2 above, with the guarantee that $\|M - U'V'\|_\infty \le O(\delta)$. Thus, in (2), the first line would be replaced by the inequality $\|M - UV\|_\infty \le \|U'V' - UV\|_\infty + O(\delta)$, and this extra error term of $O(\delta)$ will carry through in all the subsequent inequalities in (2).

Further, although these rational approximations for the rows of U' and the columns of V' may not be in the polytope P defined in Step 1 above, they will be within $O(\delta)$ distance of P.

The problem of computing spectral decompositions to within any desired accuracy was shown to be possible in time polynomial in the size of the matrix and $\log(\frac{1}{\delta})$, where $\delta > 0$ is the desired accuracy (under any matrix norm, and since for us the dimensions of these matrices are constants, i.e., $r \times r$, the choice of the norm also does not matter) [19]. This simply means that instead of projecting in to the closest point to the PSD cone, we instead project to some approximation of the closest point. However, this error can also be controlled. Note that the approximating point will also be in the PSD cone (it might just not be the closest one).

Thus, by keeping track of these additional error terms and defining the error parameters appropriately based on the given $\epsilon > 0$, we can still keep the guarantee $\|M - UV\|_\infty \leq \epsilon\|M\|_\infty$.

5 Open Questions

Question 1 remains the outstanding open question in the line of research on factorization algorithms with polynomial time guarantees. Another interesting direction would be generalize Theorem 2 to approximation guarantees with other norms. For example, the induced norms $\|M\|_{1,2} := \max_{x\in\mathbb{R}^m} \frac{\|Mx\|_2}{\|x\|_1}$ and $\|M\|_{\infty,2} := \max_{x\in\mathbb{R}^m} \frac{\|Mx\|_2}{\|x\|_\infty}$ were used in [12]. The authors show that approximate factorization with respect to these norms give rise to small SDP reformulations whose projections approximate a given polytope, where the geometric approximation is tightly determined by the approximation factor in the matrix factorization.

It would also be interesting to resolve the following question:

Let $r \in \mathbb{N}$ and $\epsilon > 0$ be fixed constants. Let $\mathcal{M}$ be the family of nonnegative matrices such that for every $M \in \mathcal{M}$, there exists another nonnegative matrix $\overline{M}$ such that $\|M - \overline{M}\|_\infty \leq \epsilon\|M\|_\infty$ and $\overline{M}$ admits a rank r PSD factorization.
Does there exists an algorithm which, given any nonnegative matrix $M \in \mathcal{M}$, can find matrices U and V such that each row of U and each column of V are in $\mathcal{S}^r_+$ such that

$$\|M - UV\|_\infty \leq O(\epsilon)\|M\|_\infty,$$

and has runtime polynomial in the dimensions of M? In other words: if the input matrix M is *close* to a matrix with small PSD rank, can we find a low PSD-rank factorization that is a good approximation to M?

Approximate low-rank nonnnegative factorizations of matrices with high nonnegative rank have been extensively studied – see [3] for a survey of the diverse applications, and [1] for a recent algorithm with provable guarantees on the complexity. The corresponding question for PSD factorizations is of similar interest.

References

1 Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization–provably. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of computing*, STOC'12, pages 145–162. ACM, 2012.

2 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM (JACM)*, 43(6):1002–1045, 1996.

3 Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

4 Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: W-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc*, 21(1):1–46, 1989.

5 Gábor Braun, Jonah Brown-Cohen, Arefin Huq, Sebastian Pokutta, Prasad Raghavendra, Aurko Roy, Benjamin Weitz, and Daniel Zink. The matching problem has no small symmetric sdp. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'16, pages 1067–1078, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=2884435.2884510`.

6 Jop Briët, Daniel Dadush, and Sebastian Pokutta. On the existence of 0/1 polytopes with high semidefinite extension complexity. *Mathematical Programming*, pages 1–21, 2014.

7 Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Extended formulations in combinatorial optimization. *4OR*, 8(1):1–48, 2010.

8 Hamza Fawzi, Jo ao Gouveia, Pablo A. Parrilo, Richard Z. Robinson, and Rekha R. Thomas. Positive semidefinite rank. *http://arxiv.org/abs/1407.4095*, 2015.

9 Samuel Fiorini, Volker Kaibel, Kanstantsin Pashkovich, and Dirk Oliver Theis. Combinatorial bounds on nonnegative rank and extended formulations. *Discrete mathematics*, 313(1):67–83, 2013.

10 Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 95–106. ACM, 2012.

11 Joao Gouveia, Pablo A Parrilo, and Rekha R Thomas. Lifts of convex sets and cone factorizations. *Mathematics of Operations Research*, 38(2):248–264, 2013.

12 João Gouveia, Pablo A Parrilo, and Rekha R Thomas. Approximate cone factorizations and lifts of polytopes. *Mathematical Programming*, 151(2):613–637, 2015.

13 D Yu Grigor'ev and NN Vorobjov. Solving systems of polynomial inequalities in subexponential time. *Journal of symbolic computation*, 5(1):37–64, 1988.

14 Didier Henrion and Jérôme Malick. Projection methods in conic optimization. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 565–600. Springer, 2012.

15 Volker Kaibel. Extended formulations in combinatorial optimization. *arXiv preprint arXiv:1104.1023*, 2011.

16 J Lee, Prasad Raghavendra, David Steurer, and Ning Tan. On the power of symmetric lp and sdp relaxations. In *Proceedings of the 29th Conference on Computational Complexity (CCC)*, pages 13–21. IEEE, 2014.

17 James R. Lee, Prasad Raghavendra, and David Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC'15, pages 567–576, New York, NY, USA, 2015. ACM. `doi:10.1145/2746539.2746599`.

18 Ankur Moitra. An almost optimal algorithm for computing nonnegative rank. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'13, pages 1454–1464. SIAM, 2013.

19 Victor Y Pan and Zhao Q Chen. The complexity of the matrix eigenproblem. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of computing*, pages 507–516. ACM, 1999.

20 Yuval Rabani and Amir Shpilka. Explicit construction of a small ϵ-net for linear threshold functions. *SIAM Journal on Computing*, 39(8):3501–3520, 2010.

21 Thomas Rothvoß. Some 0/1 polytopes need exponential size extended formulations. *Mathematical Programming*, 142(1-2):255–268, 2013.

22 Thomas Rothvoß. The matching polytope has exponential extension complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 263–272. ACM, 2014.

23 Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

24 Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 223–228. ACM, 1988.

25 G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995. `doi:10.1007/978-1-4613-8431-1`.

Hardness of Approximation for H-Free Edge Modification Problems

Ivan Bliznets[*1], **Marek Cygan**[†2], **Paweł Komosa**[†3], **and Michał Pilipczuk**[‡4]

1 **St. Petersburg Department of Steklov Institute of Mathematics, Russia**
`iabliznets@gmail.com`

2 **Institute of Informatics, University of Warsaw, Poland**
`cygan@mimuw.edu.pl`

3 **Institute of Informatics, University of Warsaw, Poland**
`p.komosa@mimuw.edu.pl`

4 **Institute of Informatics, University of Warsaw, Poland**
`michal.pilipczuk@mimuw.edu.pl`

Abstract

The H-free Edge Deletion problem asks, for a given graph G and integer k, whether it is possible to delete at most k edges from G to make it H-*free*, that is, not containing H as an induced subgraph. The H-free Edge Completion problem is defined similarly, but we add edges instead of deleting them. The study of these two problem families has recently been the subject of intensive studies from the point of view of parameterized complexity and kernelization. In particular, it was shown that the problems do not admit polynomial kernels (under plausible complexity assumptions) for almost all graphs H, with several important exceptions occurring when the class of H-free graphs exhibits some structural properties.

In this work we complement the parameterized study of edge modification problems to H-free graphs by considering their approximability. We prove that whenever H is 3-connected and has at least two non-edges, then both H-free Edge Deletion and H-free Edge Completion are very hard to approximate: they do not admit poly(OPT)-approximation in polynomial time, unless $\mathrm{P} = \mathrm{NP}$, or even in time subexponential in OPT, unless the Exponential Time Hypothesis fails. The assumption of the existence of two non-edges appears to be important: we show that whenever H is a complete graph without one edge, then H-free Edge Deletion is tightly connected to the Min Horn Deletion problem, whose approximability is still open. Finally, in an attempt to extend our hardness results beyond 3-connected graphs, we consider the cases of H being a path or a cycle, and we achieve an almost complete dichotomy there.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases hardness of approximation, parameterized complexity, kernelization, edge modification problems

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.3

[*] The research of I. Bliznets is supported by the Government of the Russian Federation (grant 14.Z50.31.0030), by the Grant of the President of the Russian Federation (MK-6550.2015.1) as well as by Warsaw Center of Mathematics and Computer Science.

[†] The work of M. Cygan and P. Kamosa is the part of a project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677651).

[‡] The research of M. Pilipczuk is supported by Polish National Science Centre grant UMO-2013/11/D/ST6/03073. M. Pilipczuk is also supported by the Foundation for Polish Science (FNP) via the START stipend programme.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016).
Editors: Klaus Jansen, Claire Matthieu, José D. P. Rolim, and Chris Umans; Article No. 3; pp. 3:1–3:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

We consider the following general setting of *graph modification problems*: given a graph G, one would like to modify G as little as possible in order to make it satisfy some fixed property of global nature. Motivated by applications in de-noising data derived from imprecise experimental measurements, graph modification problems occupy a prominent role in the field of parameterized complexity and kernelization. This is because the allowed number of modifications usually can be assumed to be small compared to the total instance size, which exactly fits the motivation of considering it as the parameter of the instance.

Moving to the formal setting, consider some hereditary class of graphs Π, that is, a class closed under taking induced subgraphs. For such a class Π, we can define several problems depending on the set of allowed modifications. In each case the input consists of a graph G and integer k, and the question is whether one can apply at most k modification to G so that it falls into class Π. In this paper we will consider *deletion* and *completion* problems, where we are allowed only to delete edges, respectively only to add edges. However, other studied variants include *vertex deletion* problems (the allowed modification is removal of a vertex) and *editing* problems (both edge deletions and completions are allowed). Moreover, we restrict ourselves to classes Π characterized by one forbidden induced subgraph H. In other words, Π is the class of H-free graphs, that is, graphs that do not contain H as an induced subgraph (H is assumed to be constant).

The study of the parameterized complexity of H-free Edge Deletion and H-free Edge Completion focused on two aspects: designing fixed-parameter algorithms and kernelization procedures. The classic observation of Cai [3] shows that H-free Edge Deletion (Completion) can be both solved in time $c^k \cdot n^{\mathcal{O}(1)}$ for some constant c depending only on H, using a straightforward branching strategy. However, for several completion problems related to chordal graphs and their subclasses, like (proper) interval graphs or trivially perfect graphs, one can design *subexponential parameterized algorithms*, typically with the running time of $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$. The study of this surprising subexponential phenomenon, and of its limits, has recently been the subject of intensive studies; we refer to the introductory section of [2] for more details. However, for the vast majority of graphs H, the running time of the form $c^k \cdot n^{\mathcal{O}(1)}$ is essentially the best one can hope for H-free Edge Deletion (Completion). Indeed, Aravind et al. [1] proved that, whenever H has at least two edges, then H-free Edge Deletion is NP-hard and has no $2^{o(k)} \cdot n^{\mathcal{O}(1)}$-time algorithm unless the Exponential Time Hypothesis fails, and the same result holds for H-free Edge Completion whenever H has at least two non-edges. The remaining cases are easily seen to be polynomial-time solvable, so this establishes a full dichotomy.

Another interesting aspect of graph modification problems is their kernelization complexity. Recall that a *polynomial kernel* for a parameterized problem is a polynomial-time algorithm that, given an instance of the problem with parameter k, reduces it to another instance of the same problem that has size bounded polynomially in k. While every H-Free Vertex Deletion problem admits a simple polynomial kernel by a reduction to the d-Hitting Set problem (for $d = |V(H)|$), the situation for edge deletion and edge completion problems is much more complex. This is because the removal/addition of some edge may create new induced copies of H that were originally not present, and hence the obstacles can "propagate" in the graph. In fact, a line of work [4, 5, 9, 12] showed that, unless NP $\subseteq$ coNP/poly, polynomial kernels for the H-free Edge Deletion (Completion) problems exist only for very simple graphs H, for which the class of H-free graphs exhibits some structural property. This line culminated in the work of Cai and Cai [4, 5], who attempted to obtain a complete dichotomy. While this goal was not fully achieved and there are some cases missing, the obtained complexity picture explains the general situation very well. For example, Cai and Cai [4, 5] showed that polynomial kernels do not exist (under NP $\not\subseteq$ coNP/poly) for the H-free Edge Deletion (Completion) problems whenever H is 3-connected and has at least 2 non-edges. Nontrivial positive cases include e.g. H being a path on 4 vertices [9] (that is, Cograph Edge Deletion (Completion)), and H being a K_4 minus one edge [5] (that is, Diamond-free Edge Deletion). One of the most prominent open cases left is the kernelization complexity of Claw-Free Edge Deletion [4, 6].

Our motivation and results

The starting point of our work is the realization that the propagational character of H-free Edge Deletion (Completion), which is the basic explanation of its apparent kernelization hardness, also makes the greedy approach to approximation incorrect. One cannot greedily remove all the edges of any copy of H in the graph, because removing an edge does not necessarily always help: it may create new copies of H in the instance. Hence, the approximation complexity of H-free Edge Deletion (Completion) is actually also highly unclear. On the other hand, the links between approximation and kernelization are well-known in parameterized complexity: it is often the case that a polynomial kernel for a problem can be turned into a poly(OPT)-approximation algorithm (i.e. an algorithm that returns a solution of cost bounded by some polynomial function of the optimum), by just taking greedily the kernel and reverting the reduction rules. While this intuitive link is far from being formal, and actually there are examples of problems behaving differently [8], it is definitely the case that the combinatorial insight given by kernelization algorithms may be very useful in the approximation setting.

Therefore, we propose to study the approximability of H-free Edge Deletion (Completion) as well, alongside with the best possible running times of fixed-parameter algorithms and the existence of polynomial kernels. This work is the first step in this direction.

We prove that the H-free Edge Deletion (Completion) problems are very hard to approximate for a vast majority of graphs H, which mirrors the kernelization hardness results of Cai and Cai [4, 5]. The following theorem explains our main result formally.

▶ **Theorem 1.** *Let H be a 3-connected graph with at least two non-edges. Then, unless* P = NP*, neither H-free Edge Deletion nor H-free Edge Completion admits a* poly(OPT)*-approximation algorithm running in polynomial time. Moreover, unless the Exponential Time Hypothesis fails, neither of these problems admits even a* poly(OPT)*-approximation algorithm running in time $2^{o(\mathsf{OPT})} \cdot n^{\mathcal{O}(1)}$.*

Theorem 1 makes two structural assumptions about graph H: that it is 3-connected, and has at least two non-edges. The first one is a crucial technical ingredient in the reductions, because it enables us to argue that for any vertex cut of size 2, every copy of H in the graph is completely contained on one side of the cut. Relaxing this assumption is a major issue addressed by Cai and Cai [4, 5] in their work. In an attempt to lift this assumption in our setting as well, we try to resolve the case of H being a path or a cycle first; this reflects the development of the story of kernelization hardness for the considered problems [5, 4, 9, 12]. The following theorem summarizes our results in this direction.

▶ **Theorem 2.** *Let H be a cycle on at least 4 vertices or a path on at least 5 vertices. Then, unless* P = NP*, neither H-free Edge Deletion nor H-free Edge Completion admits a* poly(OPT)*-approximation algorithm running in polynomial time. Moreover, unless the*

Exponential Time Hypothesis fails, neither of these problems admits even a poly(OPT)*-approximation algorithm running in time* $2^{o(\mathsf{OPT})} \cdot n^{\mathcal{O}(1)}$.

Together with some easy cases and known positive results [14], this gives an almost complete dichotomy for paths and cycles. The only missing case is Cograph Edge Deletion (for $H = P_4$), for which we expect a positive answer due to the existence of a polynomial kernel [9]. However, our preliminary attempt at lifting the kernel of Guillemot et al. [9] showed that the approach does not directly work for approximation, and new insight seems to be necessary.

Finally, somewhat surprisingly we show that the assumption that H has at least two non-edges appears to be important. Suppose $H = K_n \setminus e$ is a complete graph on $n \geq 5$ vertices with one edge removed. While H-free Edge Completion is trivially polynomial-time solvable, due to each obstacle having only one way to be destroyed, the complexity of H-free Edge Deletion turns out to be much more interesting. Namely, we show that it is tightly connected to the complexity of Min Horn Deletion, which apparently is one of the remaining open cases in the classification of the approximation complexity of CSP problems of Khanna et al. [11]. Hence, the following theorem shows that the case of H being a complete graph without an edge may be an interesting outlier in the whole complexity picture.

▶ **Theorem 3.** *For any $n \geq 5$, the $K_n \setminus e$-free Edge Deletion problem is* Min Horn Deletion*-complete with respect to A-reductions.*

The exact meaning of Min Horn Deletion-completeness, A-reductions and other definitions related to the hardness of approximation for CSP problems are explained in Section 4. A direct consequence of Theorem 3 and the work of Khanna et al. [11] is that $K_n \setminus e$-free Edge Deletion does not admit a $2^{\mathcal{O}(\log^{1-\epsilon}|E|)}$-approximation algorithm working in polynomial time, for any $\epsilon > 0$, where $|E|$ is the number of edges in a given graph. Moreover, Theorem 3 implies that $K_n \setminus e$-free Edge Deletion is poly-APX-hard if and only if each Min Horn Deletion-complete problem is poly-APX-hard, the latter being an intriguing open problem left by Khanna et al. [11] in their study of approximability of CSPs.

While there is no direct connection between the existence of a poly(OPT) approximation and poly-APX-hardness, we still believe that our reduction corroborates the hardness of resolving approximation question of $K_n \setminus e$-free Edge Deletion in terms of optimum value. Intuitively, showing poly-APX-hardness should be easier than refuting poly(OPT) approximation. Below we state formally what our reduction actually implies.

▶ **Corollary 4.** *Let $n \geq 5$. Then it is* NP*-hard to approximate the $K_n \setminus e$-free Edge Deletion problem within factor $2^{\mathcal{O}(\log^{1-\epsilon}|E|)}$ for any $\epsilon > 0$, where $|E|$ is the number of edges in a given graph.*

▶ **Corollary 5.** *Let $n \geq 5$. Then the $K_n \setminus e$-free Edge Deletion problem admits an n^{δ}-approximation for all $\delta > 0$, if and only if each* Min Horn Deletion*-complete problem admits an n^{δ_1}-approximation for all $\delta_1 > 0$.*

Our techniques

To prove our main result, Theorem 1, we employ the following strategy. We first consider the *sandwich problem* defined as follows: in Sandwich H-Free Edge Deletion we are given a graph G together with a subset D of *undeletable edges*, and the question is whether there exists a subset $F \subseteq E(G) \setminus D$ of deletable edges for which $G - F$ is H-free. Note that the sandwich problem differs from the standard H-free Edge Deletion problem in two aspects: first, some edges are forbidden to be deleted, and, second, it is a decision problem about the existence of *any* solution—we do not impose any constraint on its size. For completion, the sandwich problem is defined similarly: we have *non-fillable non-edges*, i.e., non-edges that are forbidden to be added in the solution.

The crux of the approach is to prove that Sandwich H-Free Edge Deletion is actually NP-hard under the given assumptions on H. The next step is to reduce from the sandwich problem to the standard optimization variant. This is done by adding gadgets that emulate undeletable edges by introducing a large approximation gap, as follows. For each undeletable edge e, attach a large number of copies of H to e, so that each copy becomes an induced H-subgraph if e gets deleted. Then any solution that deletes the undeletable edge e must have a very large cost, due to all the disjoint copies of H that appear after the removal of e. The assumption that H is 3-connected is very useful for showing that the constructions do not introduce any additional, unwanted copies of H in the graph.

The approach for completion problems is similar. To prove Theorem 2 that concerns paths and cycles, we give problem-specific constructions using the same approach. Some of them are based on previous ETH-hardness proofs for the problems, given by Drange et al. [7].

As far as Theorem 3 is concerned, we employ a similar reduction strategy, but instead of starting from 3SAT, we start from a carefully selected MinOnes($\mathcal{F}$) problem: the problem of optimizing the number of ones in a satisfying assignment to a boolean formula that uses only constraints from some fixed family $\mathcal{F}$. In particular, the constraint family $\mathcal{F}$ needs to be rich enough to be Min Horn Deletion-hard, while at the same time it needs to restrictive enough so that it can be expressed in the language of $K_n \setminus e$-free Edge Deletion.

Our constructions are inspired by the rich toolbox of hardness proofs for kernelization and fixed-parameter algorithms for edge modification problems [1, 4, 5, 7, 12, 9]. In particular, the idea of considering sandwich problems can be traced back to the work of Cai and Cai [4, 5], who use the term *quarantine* for the optimization variants of sandwich edge modification problems, with undeletable edges and non-fillable non-edges. Quarantined problems serve a technical, auxiliary role in the work of Cai and Cai [4, 5]: one first proves hardness of the quarantined problem, and then lifts the quarantine by attaching gadgets, similarly as we do.

However, we would like to point out the new challenges that appear in the approximation setting. Most importantly, the vast majority of previous reductions heavily use budget constraints (i.e. the fact that the solution is stipulated to be of size at most k) to argue the correctness; this includes the general results of Cai and Cai [4, 5]. In our setting, we cannot use arguments about the tightness of the budget, because we need to introduce a large approximation gap at the end of the construction. The usage of the sandwich problems *without* any budget constraints is precisely the way we overcome this difficulty. Thus, most of the old reductions do not work directly in our setting, but of course some technical constructions and ideas can be salvaged.

Outline

In Section 2 we introduce terminology and recall the most important facts from the previous works. Section 3 is devoted to the proof of our main result, Theorem 1. However, as the proof for H-free Edge Completion is similar to the proof for H-free Edge Deletion, the details are postponed to the full version of the paper (arXiv:1606.02688). In Section 4 we discuss the proof of Theorem 3. The proof of Theorem 2 is also omitted, and included in the full version of the paper. Concluding remarks and prospects on future work are in Section 5.

2 Preliminaries

2.1 Basic graph definitions

We use standard graph notation. For a graph G by $V(G)$ and $E(G)$ we denote the set of vertices and edges of G, respectively. Throughout the paper we consider simple graphs only, i.e., there are no self-loops nor parallel edges. We use K_n to denote the complete graph on n vertices. By P_ℓ (C_ℓ) we denote the path (cycle) with exactly ℓ vertices. By $\overline{G}$ we denote the *complement* of G, i.e., a graph on the same vertex set, where two distinct vertices are adjacent if and only if they were not adjacent in G. We say that a graph G is H-free, if G does not contain H as an induced subgraph.

We define a graph G to be 3-*vertex-connected* if G has at least 3 vertices, and removing any set of at most two vertices causes G to stay connected. For brevity, we call such graphs 3-*connected*.

2.2 Problems and approximation algorithms

In the decision version the H-free Edge Deletion (Completion) problem, for a given graph G and an integer k, one is to decide whether it is possible to delete (add) at most k edges from (to) G to make it H-free. In particular, we consider the $\overline{P}_5$-Free Deletion (Completion) problem, and call it House-Free Deletion (Completion). However, in the optimization variant of H-free Edge Deletion (Completion) the value of k is not given and the goal is to find a minimum size solution. It will be clear from the context whether we refer to a decision or optimization variant.

In the Sandwich H-Free Edge Deletion (Completion) problem we are given a graph G together with a subset D of *undeletable edges* (*non-fillable non-edges*). The question is whether there exists a subset $F \subseteq E(G) \setminus D$ ($F \subseteq \overline{E(G)} \setminus D$) of deletable (fillable) edges for which $G - F$ ($G + F$) is H-free. Note that it is a decision problem, where we ask about existence of any solution, i.e., we do not impose any constraint on the solution size.

Let f be a fixed non-decreasing function on positive integers. An $f(OPT)$-*factor approximation algorithm* for a minimization problem X is an algorithm that finds a solution of size at most $f(OPT) \cdot OPT$, where OPT is the size of an optimal solution for a given instance of X.

2.3 Satisfiability and Exponential Time Hypothesis

We employ the standard notation related to satisfiability problems. A 3CNF formula is a conjunction of clauses, where a clause is a disjunction of at most three literals. The 3SAT problem asks, for a given formula φ, whether there is a satisfying assignment to φ.

The Exponential Time Hypothesis (ETH), introduced by Impagliazzo, Paturi and Zane [10] is now an established tool used for proving conditional lower bounds in the parameterized complexity area (see [13] for a survey on ETH-based lower bounds).

▶ **Hypothesis 6** (Exponential Time Hypothesis (ETH) [10]). *There is no $2^{o(n)}$ time algorithm for* 3SAT*, where n is the number of variables of the input formula.*

The main consequence of the Sparsification Lemma of [10] is the following theorem: there is no subexponential algorithm for 3SAT even in terms of the number of clauses of the formula.

▶ **Theorem 7** ([10]). *Unless ETH fails, there is no $2^{o(n+m)}$ time algorithm for* 3SAT*, where n, m are the number of variables, and clauses, respectively.*

3 Hardness for 3-connected H

In this section we present the proof of Theorem 1 for H-free Edge Deletion.

3.1 Deletion problems

We start with proving hardness of the sandwich problem.

▶ **Lemma 8.** *Let H be a 3-connected graph with at least 2 non-edges. There is a polynomial-time reduction, which given an instance of* 3SAT *with n variables and m clauses, creates an equivalent instance of* Sandwich H-free Edge Deletion *with $\mathcal{O}(n+m)$ edges. Consequently,* Sandwich H-free Edge Deletion *is* NP*-hard for such graphs H.*

Proof. Let φ be the given formula in 3CNF, and let `vars` and `cls` be the sets of variables and clauses of φ. By standard modifications of the formula, we may assume that each clause contains exactly three literals of pairwise different variables. We construct an instance G of Sandwich H-free Edge Deletion as follows. The graph G is created from three types of gadgets: a clause gadget, a variable gadget, and a connector gadget. They are depicted in Figure 1, where presented edges are deletable, and all others are undeletable.

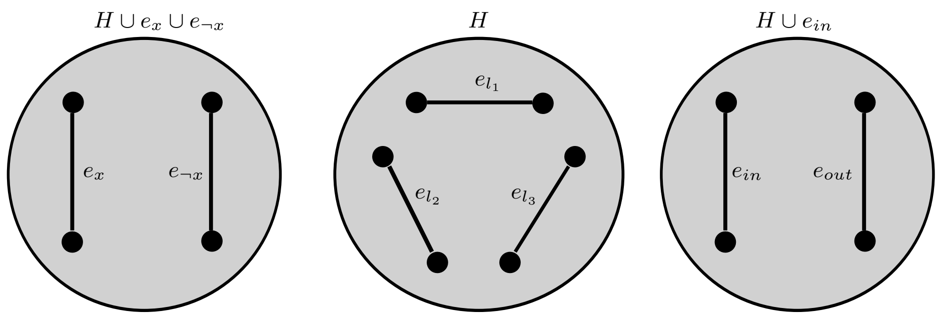
We first explain constructions of the gadgets, and then discuss connections between them. For each variable $x \in$ `vars`, we create a variable gadget G^x, which is the graph H with two added edges e_x and $e_{\neg x}$ in place of any two non-edges of H. In the graph H_x, all edges are marked as undeletable except e_x and $e_{\neg x}$. Intuitively, deletion of the edge e_x or $e_{\neg x}$ mimics an assignment of the corresponding literal to true. The variable gadget forbids simultaneous assignments of both literals to true. If we delete both edges e_x and $e_{\neg x}$, we get an induced subgraph H in which we cannot delete any edge.

Each clause $c = \ell_1 \vee \ell_2 \vee \ell_3 \in$ `cls` has the corresponding clause gadget H^c, which is a copy of the graph H. As H^c is 3-connected, it has at least 3 edges. We pick arbitrarily three edges of H^c and label them by $e_{\ell_1}, e_{\ell_2}, e_{\ell_3}$. We mark all others edges as undeletable. In order to make the clause gadget H-free, we have to delete at least one edge from $e_{\ell_1}, e_{\ell_2}, e_{\ell_3}$ (note that some of the three distinguished edges might potentially share an endpoint). Intuitively, deletion of the edge labeled by e_ℓ corresponds to assigning value true to literal ℓ.

The third type of gadgets is the connector gadget. The connector gadget C is a copy of the graph H, with one added edge in place of any non-edge of H. We label this edge as e_{in}. In C, there also exists another edge that does not share any of its endpoints with e_{in}. To see this, for the sake of contradiction suppose that every edge of C is incident to one of the endpoints of e_{in}. If C has at least two vertices other than these endpoints, then the endpoints of e_{in} form a vertex cut of size 2 separating them, a contradiction with 3-connectedness of H. Otherwise C has only one vertex other than the endpoints of e_{in}, so H has at most 3 vertices; again, a contradiction with the 3-connectedness of H, as we assume H to have at least 2 non-edges. We select any edge in H that does not share endpoints with e_{in}, and we label it as e_{out}. Edges e_{in} and e_{out} are made deletable, and all other edges of C are made undeletable. Note that deletion of the edge e_{in} creates an induced subgraph H, and then we have to delete e_{out} in order to destroy this subgraph.

Knowing the structure of all gadgets, we can proceed with the main construction of our reduction.

Given a formula φ, for each clause $c \in$ `cls` and variable $x \in$ `vars`, we create the clause gadget H^c and the variable gadget G^x, respectively. Moreover, for each literal ℓ belonging to the clause $c \in$ `cls`, we create a chain $C_1^{\ell,c}, C_2^{\ell,c}, \ldots, C_{p+2}^{\ell,c}$ consisting of $p+2$ copies of the connector gadget, where $p = |V(H)|$. This chain is constructed in the following way: the



(a) Variable gadget G^x **(b)** Clause gadget H^c **(c)** Connector gadget C

Figure 1 Gadgets for Sandwich H-free Edge Deletion.

edge e_{out} of $C_i^{\ell,c}$ is identified with the edge e_{in} of $C_{i+1}^{\ell,c}$, for $i = 1, \dots, p+1$. We also identify the edge e_{out} in the subgraph $C_{p+2}^{\ell,c}$ with the edge e_ℓ in the variable gadget of the variable of ℓ. Moreover, the edge e_{in} in the subgraph $C_1^{\ell,c}$ is identified with the edge e_ℓ from the clause gadget H^c. We use those chains to not allow the copy of H to be shared by any two gadgets, and we will prove it in the claim below.

Clearly, the constructed graph G has at most $\mathcal{O}(n+m)$ edges.

▶ **Claim 9.** *If G is a YES instance, then φ is satisfiable.*

Proof. Take any solution to the instance G. Note that in each clause gadget we must delete at least one edge. We set the literals corresponding to the deleted edges to true, thus satisfying every clause. We prove now that for each variable x we have not set both literals x and $\neg x$ to true, so that we can find a true/false assignment to the variables that sets the literals accordingly. Deletion of an edge in the clause gadget propagates deletions up to the variable gadget via the chain of connector gadgets. This happens because the deletion of e_{in} in $C_1^{\ell,c}$ forces us to delete the e_{out} in $C_1^{\ell,c}$, which is e_{in} in $C_2^{\ell,c}$, so we are forced to delete e_{out} in $C_2^{\ell,c}$, and so on. Following the chain of connector gadgets, it is easy to see that the edge e_ℓ must be deleted in the corresponding variable gadget. As the solution to the instance G cannot delete both edges e_x and $e_{\neg x}$ in any variable gadget at the same time, we obtain that there are no variables with both of its literals set to true. ◀

▶ **Claim 10.** *If φ is satisfiable, then G is a YES instance.*

Proof. Consider a true/false assignment that satisfies the formula φ and delete all edges in all clause gadgets that correspond to literals taking value true. Propagate deletions to all the connector and variable gadgets, as in the proof of Claim 9. It remains to prove that the obtained graph is indeed an H-free graph. By counting the number of edges in each gadgets, it follows that after the deletions, all gadgets become not isomorphic to H: in every variable gadget, we deleted exactly one edge, in every clause gadget, we deleted at least one edge, and in each connector gadget we deleted zero or two edges. So if the obtained graph contains an induced subgraph of H, then H is distributed across several gadgets. However, this is also not possible for the following reason.

For the sake of contradiction, suppose after the deletions there is an induced copy H' of the graph H. Since H' is connected and is distributed among more than one gadget, there have to be two different gadgets G_1, G_2 that share a vertex, for which H' contains both some vertex $u \in V(G_1) \setminus V(G_2)$, and some vertex $v \in V(G_2) \setminus V(G_1)$. Since H' is 3-connected, there are 3 internally vertex-disjoint paths in H' that lead from u to v. But

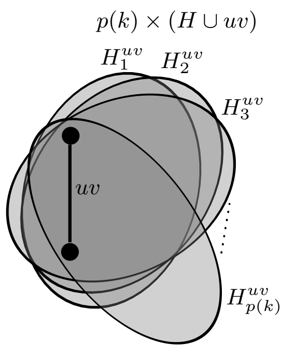


Figure 2 Gadgets H_i^{uv} for H-free Edge Deletion.

every two gadgets share at most two common vertices, so at least one of these paths, say P, avoids $V(G_1) \cap V(G_2)$. Since the path P avoids $V(G_1) \cap V(G_2)$, from the construction of G it easily follows that such path P contains at least one vertex of some variable gadget and at least one vertex of some clause gadget. However, the distance between e_{in} and e_{out} in each connector gadget is at least 1, so the distance between any variable gadget and any clause gadget is at least $|V(H)|$. But the path P is entirely contained in H', thus its length is at most $|V(H)| - 1$, a contradiction. ◀

Claims 9 and 10 ensure that the output instance G is equivalent to the input instance φ of 3SAT, so we are done. ◀

Now, we show how to reduce Sandwich H-free Edge Deletion to the optimization variant of H-free Edge Deletion. Note that we only require H to have at least one non-edge; this is because we will reuse this lemma in the next section.

▶ **Lemma 11.** *Let H be a 3-connected graph with at least one non-edge, and $p(\cdot)$ be a polynomial with $p(\ell) \geq \ell$, for all positive ℓ. Then there is a polynomial-time reduction which, given an instance G of Sandwich H-free Edge Deletion, creates an instance (G', k) of H-free Edge Deletion, such that:*
- *k is the number of deletable edges in G;*
- *G' has $\mathcal{O}(p(k) \cdot |E(G)| \cdot |E(H)|)$ edges;*
- *If G is a YES instance, then (G', k) is a YES instance;*
- *If G is a NO instance, then $(G', p(k))$ is a NO instance.*

Proof. We create G' in the following way. For each undeletable edge uv, we add $p(k)$ copies H_i^{uv} of the graph H, $i = 1, \ldots, p(k)$. In each copy, we choose any non-edge u_iv_i and identify the vertex u_i with u, and v_i with v. The construction is presented in Figure 2.

Note that if we delete the edge uv in G', we also must delete at least one edge in every H_i^{uv}. Hence, at least $p(k) + 1$ edges will be deleted in such a situation. With this observation in mind, we proceed to the proof of the correctness.

▶ **Claim 12.** *If G is a YES instance, then (G', k) is a YES instance.*

Proof. Let F be a subset deletable edges, such that $G - F$ is H-free. Obviously $|F| \leq k$, because there are k deletable edges in G in total. We will prove that $G' - F$ is also H-free, which implies that (G', k) is a YES instance.

Let us assume otherwise, that there is an induced copy H' of H in G'. Since $G - F$ is H-free, we have that H' has to contain at least one vertex of $V(G') \setminus V(G)$. Say that H'

contains some vertex x of $V(H_i^{uv}) \setminus V(G)$, for some undeletable edge uv and some index i. The edge uv is undeletable in G, so it is not included in F. Consequently, the subgraph of G' induced by $V(H_i^{uv})$ contains one more edge than H, so it is not isomorphic to H. We conclude that H' must contain some vertex y that lies outside of $V(H_i^{uv})$. Since H is 3-connected, there are 3 internally vertex-disjoint paths between x and y in H. However, in G, the set $V(H_i^{uv}) \cap V(G) = \{u, v\}$ is a vertex cut of size 2 that separates x and y. This is a contradiction, so $G' - F$ is indeed H-free. ◀

▶ **Claim 13.** *If G is a NO instance, then $(G', p(k))$ is a NO instance.*

Proof. For the sake of contradiction, suppose there is a set F' of at most $p(k)$ edges of G', such that $G' - F'$ is H-free. Note that, F' has to contain at least one undeletable edge uv, as otherwise $F' \cap E(G)$ would be a solution to G. But then F' has to contain at least $p(k)$ more edges inside gadgets H_i^{uv}, for $i = 1, 2, \ldots, p(k)$, which is a contradiction with $|F'| \leq p(k)$. ◀

Claims 12 and 13 ensure the correctness of the reduction, and hence we are done. ◀

By composing the reductions of Lemmas 8 and 11, we can deduce the part of Theorem 1 concerning deletion problems. Indeed, suppose H-free Edge Deletion admitted a polynomial-time $q(\mathsf{OPT})$-factor approximation algorithm, for some polynomial q. Take any instance of 3SAT, and apply first the reduction of Lemma 8, and then the reduction of Lemma 11 for polynomial $p(\ell) = q(\ell) \cdot \ell + 1$. Finally, observe that the application of the hypothetical approximation algorithm for H-free Edge Deletion to the resulting instance would resolve whether the optimum value is at most k or at least $p(k)$, which, by Lemma 11, resolves whether the input instance of 3SAT is satisfiable. The subexponential hardness of approximation under ETH follows from the same reasoning and the observation that the value of k in the output instance is bounded linearly in the size of the input formula.

4 Connections with Min Horn Deletion

In this section we prove Theorem 3. First, we need to introduce some definitions and notation regarding Min Horn Deletion hardness and completeness.

Khanna et al. [11] attempted to establish a full classification of approximability of boolean constraint satisfaction problems. In particular, many problems have been classified as APX-complete or poly-APX-complete. Even though some cases remained unresolved, Khanna et al. [11] grouped them into classes, such that all problems from the same class are equivalent (with respect to appropriately defined reductions) to a particular representative problem. One such representative problem is Min Horn Deletion, defined as follows: Given is a boolean formula φ in CNF that contains only unary clauses, and clauses with three literals out of which exactly one is negative. The problem asks for minimizing the number of ones in a satisfying assignment for φ.

We are not going to operate on instances of Min Horn Deletion directly, so the definition above is given only in order to complete the picture for the reader. Instead, we will rely on the approximation hardness results exhibited by Khanna et al. [11], which relate the approximability of various boolean CSPs to Min Horn Deletion. In particular, it is known that Min Horn Deletion does not admit a $2^{\mathcal{O}(\log^{1-\epsilon} n_{\text{vars}})}$ approximation algorithm, unless $\mathrm{P} = \mathrm{NP}$, where n_{vars} is the number of variables in the instance. On the other hand, it is an open problem whether any Min Horn Deletion-complete problem (under A-reductions, defined below) is actually poly-APX-complete.

▶ **Definition 14** (A-reducibility, Definition 2.6 of [11])**.** A combinatorial optimization problem is said to be an NPO problem if instances and solutions can be recognized in polynomial time, solutions are polynomially-bounded in the input size, and the objective function can be computed in polynomial time from an instance and a solution.

An NPO problem P is said to be A-*reducible* to an NPO problem Q, denoted $P \leq_A Q$, if there are two polynomial-time computable functions F and G and a constant α, such that:

1. For any instance $\mathcal{I}$ of P, $F(\mathcal{I})$ is an instance of Q.
2. For any instance $\mathcal{I}$ of P and any feasible solution $\mathcal{S}'$ for $F(\mathcal{I})$, $G(\mathcal{I}, \mathcal{S}')$ is a feasible solution for $\mathcal{I}$.
3. For any instance $\mathcal{I}$ of P and any $r \geq 1$, if $\mathcal{S}'$ is an r-approximate solution for $F(\mathcal{I})$, then $G(\mathcal{I}, \mathcal{S}')$ is an (αr)-approximate solution for $\mathcal{I}$.

Intuitively, A-reductions preserve approximability problems up to a constant factor (or higher). As a source of Min Horn Deletion-hardness we will use the MinOnes($\mathcal{F}$) problem, defined below, for a particular choice of the family of constraints $\mathcal{F}$.

In the MinOnes($\mathcal{F}$) problem, we are given a ground set of boolean variables X together with a set of boolean constraints. Each constraint f is taken from a specified family $\mathcal{F}$, and f is applied to some tuple of variables from X. The goal of the problem is to find an assignment satisfying all the constraints, while minimizing the number of variables set to one. Note that the family $\mathcal{F}$ is considered a part of the problem definition, not part of the input. In order to use known results for the MinOnes($\mathcal{F}$) problem we need to define some properties of boolean constraints.

- A boolean constraint f is called *weakly positive* if it can be expressed using a CNF formula that has at most one negated variable in each clause.
- A boolean constraint f is 0-*valid* if the all-zeroes assignment satisfies it.
- A boolean constraint f is IHS-B^+ if it can be expressed using a CNF formula in which the clauses are all of one of the following types: $x_1 \vee \cdots \vee x_k$ for some positive integer $k \leq B$, or $\neg x_1 \vee x_2$, or $\neg x_1$. IHS-B^- constraints are defined analogously, with every literal being replaced by its complement.

The definition can be naturally extended to families of constraints, e.g., a family of constraints is weakly positive if all its constraints are weakly positive. We say that a family of constraints is IHS-B if it is either IHS-B^+ or IHS-B^- (or both). The following result was proved by Khanna et al. [11].

▶ **Theorem 15** (Lemmas 8.7 and 8.14 from [11])**.** *If a family of constraints $\mathcal{F}$ is weakly positive, but it is neither 0-valid nor IHS-B for any constant B, then the problem* MinOnes($\mathcal{F}$) *is* Min Horn Deletion*-complete under A-reductions; that is, there is an A-reduction from* Min Horn Deletion *to* MinOnes($\mathcal{F}$) *and an A-reduction from* MinOnes($\mathcal{F}$) *to* Min Horn Deletion. *Consequently, it is NP-hard to approximate* MinOnes($\mathcal{F}$) *within factor $2^{\mathcal{O}(\log^{1-\epsilon} n_{\mathrm{vars}})}$ for any $\epsilon > 0$, where n_{vars} is the number of variables in the given instance.*

Our strategy for the proof of Theorem 3 is as follows. In Section 4.1 we show a reduction from MinOnes($\mathcal{F}$) to a properly defined quarantined version of $K_n \backslash e$-free Edge Deletion. Next, in Section 4.2 we show a reduction which removes the quarantine. Finally, in Section 4.3 we conclude the proof of Theorem 3 and show the completeness with respect to A-reductions.

Note that having Theorem 3, we can immediately infer Corollaries 4,5 using Theorem 15 and the definition of an A-reduction.

4.1 From MinOnes (F) to Quarantined H-free Edge Deletion

In the Quarantined H-free Edge Deletion problem we are given a graph G, some edges of which are marked as undeletable. Quarantined H-free Edge Deletion is an optimization problem, where the goal is to obtain an H-free graph by removing the minimum number of deletable edges.

Next, we define the family of constraints that will be used in the MinOnes($\mathcal{F}$) problem.

▶ **Definition 16.** We define the following constraints:

- a constraint $f_1(x_1, x_2, x_3)$, which is equal to zero if and only if exactly one of the variables x_1, x_2, x_3 is set to 1;
- a constraint $f_2(x) = x$.

The family of constraints $\mathcal{F}'$ is defined as $\mathcal{F}' = \{f_1, f_2\}$.

A direct check, presented below, verifies that $\mathcal{F}'$ has the properties needed to claim, using Theorem 15, that MinOnes($\mathcal{F}'$) is Min Horn Deletion-hard.

▶ **Lemma 17.** *The family of constraints $\mathcal{F}' = \{f_1, f_2\}$ is weakly positive, and at the same time it is neither* 0*-valid, nor IHS-B for any B.*

Proof. Note that f_1 is weakly positive since $f_1(x_1, x_2, x_3) = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$. Constraint f_2 is clearly weakly positive by definition. As f_2 is not 0-valid, we have that $\mathcal{F}'$ is not 0-valid either.

We prove now that f_1 is not IHS-B for any B. First, observe that any CNF formula expressing f_1 cannot contain a clause with only positive literals, as such a clause would not be satisfied by the assignment $x_1 = x_2 = x_3 = 0$, which in turn satisfies f_1. Similarly, no clause can have only negative literals. Due to the definition of IHS-B, the only remaining case is a 2-clause with one positive and one negative literal. Without loss of generality, consider a clause $x_1 \vee \neg x_2$. Observe, that it is not satisfied by the assignment $x_1 = 0$, $x_2 = x_3 = 1$, which however satisfies f_1. Therefore f_1, and consequently $\mathcal{F}'$, is not IHS-B for any B. ◀

Consequently, Theorem 15 and Lemma 17 together imply that MinOnes($\mathcal{F}'$) is Min Horn Deletion-hard under A-reductions. We now give our main reduction, from MinOnes($\mathcal{F}'$) to Quarantined $K_n \setminus e$-free Edge Deletion.

▶ **Lemma 18.** *Let $n \geq 5$. There is a polynomial-time computable transformation T which, given an instance $\mathcal{I}$ of the* MinOnes($\mathcal{F}'$) *problem, outputs an instance $T(\mathcal{I})$ of the* Quarantined $K_n \setminus e$-free Edge Deletion *problem, such that:*

- *if $\mathcal{I}$ admits a satisfying assignment with k ones, then there is a solution of cost $\Delta \cdot k$ for the instance $T(\mathcal{I})$,*
- *if $T(\mathcal{I})$ admits a solution of cost k', then there is a satisfying assignment with $\lfloor k'/\Delta \rfloor$ ones for the instance $\mathcal{I}$,*

where $\Delta = 9n_{\text{vars}}^2 + 2$ and n_{vars} is the number of variables in $\mathcal{I}$.

Proof. First, we show how to transform an instance $\mathcal{I}$ (with a formula φ) of MinOnes($\mathcal{F}'$) into an instance $T(\mathcal{I})$ (with a graph G) of Quarantined $K_n \setminus e$-free Edge Deletion. Given an instance $\mathcal{I}$, for any constraint $f_1(x, y, z)$ we create a separate clique K_n, which will be called the *constraint clique.* We arbitrarily choose three edges in the clique and label them x, y, z. Mark all edges as undeletable except edges labelled by x, y, z. Moreover, for each variable x we additionally create a clique K_n (called further the *variable clique*), and mark all edges in the clique as undeletable except two edges, which we label by x_{in}, x_{out}. The

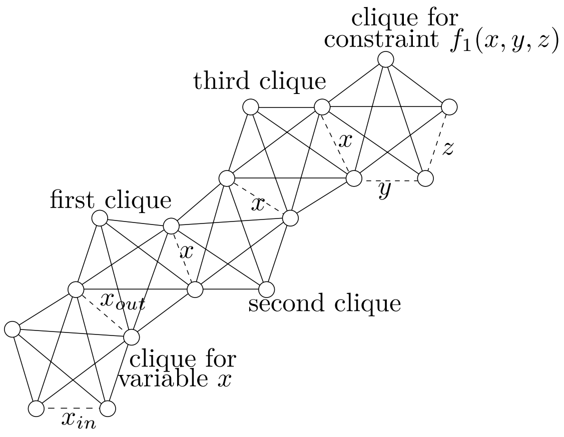


Figure 3 Gadgets for QUARANTINED $K_n \setminus e$-FREE EDGE DELETION. Deletable edges are shown by dashed lines.

edges x_{in}, x_{out} are selected arbitrarily, however we require that they do not share common endpoints.

Now we connect the variable cliques with the constraint cliques. For each variable x and a constraint f_1 of the instance $\mathcal{I}$ which contains x among its arguments, we add three cliques, as shown in Figure 3, such that the following properties are satisfied:

- The first added clique shares with the variable clique of x only the edge x_{out}.
- The second added clique shares one deletable edge with the first clique and a different deletable edge with the third clique. Label both these deletable edges by x.
- The third added clique shares with the clique corresponding to the constraint only the edge labelled (in the constraint clique) by x.

All the other edges of the introduced cliques, not mentioned above, are marked as undeletable. Note that each of the introduced cliques shares two edges with two different cliques. We may perform this construction so that these two edges never share endpoints (as depicted Figure 3), and hence we will assume this property.

Denote by $\delta(x)$ the number of occurrences of the variable x in all f_1-type constraints. Note that, by removing superfluous copies of the same constraint, we can assume that all f_1-type constraints are pairwise different, so in particular there is at most n_{vars}^3 of them. As each variable can occur in one constraint at most three times, for any variable x we have $\delta(x) \leq 3n_{\text{vars}}^2$.

Next, for each variable x we add $3\cdot(3n_{\text{vars}}^2-\delta(x))$ or $3\cdot(3n_{\text{vars}}^2-\delta(x))+1$ cliques that share the deletable edge x_{in} from the variable clique of x, and are otherwise disjoint. Moreover, in each such clique we make one more edge deletable; we label it by x. We add $3\cdot(3n_{\text{vars}}^2-\delta(x))$ cliques if the formula does contain the clause $f_2(x) = x$, and $3\cdot(3n_{\text{vars}}^2-\delta(x))+1$ cliques otherwise.

Finally, if there is a clause $f_2(x) = x$ in the instance $\mathcal{I}$, then we delete the edge labelled by x_{in} in the corresponding variable clique.

Observe that in the constructed instance of QUARANTINED $K_n \setminus e$-FREE EDGE DELETION, among all the $9n_{\text{vars}}^2 + 2$ edges labelled by x, x_{in}, x_{out}, where x is any variable, we have to delete either none, or all of them. This is because the deletion of any of them forces the deletion of all the others due to the appearance of induced copies of $K_n \setminus e$ in the graph. Moreover, if the edge x_{in} is not present due to the existence of constraint $f_2(x) = x$ in $\mathcal{I}$, then all of them have to be deleted.

▶ **Claim 19.** *If there is a satisfying assignment with k ones for the instance $\mathcal{I}$, then it is possible to delete $(9n_{\text{vars}}^2 + 2) \cdot k$ edges in $T(\mathcal{I})$ in order to make it a $K_n \setminus e$-free graph.*

Proof. It is enough to delete all edges labelled by x, x_{in}, x_{out} for all variables x that are set to 1 in the satisfying assignment; the number of such edges is exactly $(9n_{\text{vars}}^2 + 2) \cdot k$. Let us prove the statement. Suppose the obtained graph is not $K_n \setminus e$-free. Let H' be an induced subgraph isomorphic to $K_n \setminus e$. Note that for $n \geq 5$ the graph $K_n \setminus e$ is 3-connected. Moreover, even after deletion of two arbitrary vertices in $K_n \setminus e$, there are no two vertices at distance larger than two. Consequently, a direct check shows that the assumed H' subgraph must stay completely in one of the cliques corresponding to a constraint or to a variable, or in one of the cliques connecting a variable clique with a constraint clique. Obviously, H' cannot be contained in a variable clique or a connection clique, as in such cliques either all edges are present, or two edges are missing. This means that H' must stay in a constraint clique, so exactly one of the edges of this constraint clique is deleted. However, this is equivalent with the corresponding constraint being not satisfied under the considered assignment; this is a contradiction. ◀

▶ **Claim 20.** *If $T(\mathcal{I})$ admits a solution of cost k', then there is a satisfying assignment for the instance $\mathcal{I}$ with $\lfloor k'/(9n_{\text{vars}}^2 + 2) \rfloor$ ones.*

Proof. Take any solution for the output instance $T(\mathcal{I})$. As mentioned earlier, in any solution for $T(\mathcal{I})$, for any variable x either all edges labeled by x, x_{in}, x_{out} are deleted or none of them is deleted. The number of such edges for one variable x is equal to $9n_{\text{vars}}^2 + 2$. We set a variable to 1 if and only if the corresponding edges are deleted in the considered solution for $T(\mathcal{I})$. All clauses of the form $f_2(x)$ will be satisfied, since in the construction of $T(\mathcal{I})$ we delete x_{in} if the clause $f_2(x) = x$ is present in $\mathcal{I}$. All f_1-type constraints will be satisfied as well, as otherwise in the clique corresponding to an unsatisfied constraint only one edge would be deleted and, hence, the graph would not be $K_n \setminus e$-free. ◀

The correctness of the transformation follows from Claims 19 and 20; hence the proof of Lemma 18 is complete. ◀

4.2 Lifting the quarantine

In the following lemma we show how to reduce an instance of the quarantined problem to its regular version, using the same approach as in the proof of Lemma 11.

▶ **Lemma 21.** *Let $n \geq 5$. There is a polynomial-time reduction which, given an instance G of* Quarantined $K_n \setminus e$-free Edge Deletion *with m edges, outputs an instance G' of* $K_n \setminus e$-free Edge Deletion *such that:*

- *G' has $\mathcal{O}(m^3)$ vertices and edges.*
- *If there is a solution of size k for the instance G, then there is a solution of size k for the instance G'.*
- *If there is a solution of size $k \leq m^2$ for the instance G', then there is a solution of size k for the instance G.*

Proof. We apply the reduction described in the proof of Lemma 11 for $p(m) = m^2$ and $H = K_n \setminus e$. Now we verify that G' has the claimed properties. The bound on the size of G' follows directly from the size bound given by Lemma 11.

Suppose first that G has some solution of size k. In the proof of Lemma 11 we argued that the same solution also works for the instance G' (see the proof of Claim 12). Hence, G' also has a solution of size k.

Suppose now that G' has a solution F of some size $k \le m^2$. In the proof of Claim 13 we argued that F does not delete any of the undeletable edges of G, because this would require deleting at least m^2 more edges in the attached gadgets. Hence, $F \cap E(G)$ is a set of size at most k, whose deletion turns G into an H-free graph, due to being an induced subgraph of $G' - F$. Hence, G has some solution of size at most k. ◀

The composition of the reductions of Lemmas 18 and 21 gives an A-reduction (for $\alpha = 1$) from a Min Horn Deletion-hard problem MinOnes($\mathcal{F}$), yielding the hardness part of Theorem 3. Indeed, given an instance $\mathcal{I}$ of MinOnes($\mathcal{F}$) we can transform it into an instance G of Quarantined $K_n \setminus e$-free Edge Deletion using Lemma 18, which in turn we can further transform into an instance G' of $K_n \setminus e$-free Edge Deletion using Lemma 21. Given any feasible solution F' for G' we check whether $|F'| \le |E(G)|^2$. If this is the case, we translate back the solution F' into a solution F for G (using Lemma 21) and then into a solution for the initial instance $\mathcal{I}$ (using Lemma 18). On the other hand, if $|F'| > |E(G)|^2$, then we may take a trivial solution being an assignment setting all the variables to one. This is an r-approximation where $r > |E(G)|$, as $|E(G)| > n_{vars}$ for the initial instance $\mathcal{I}$. The assignment will satisfy all the contraints and will be at least an r-approximation as we need to assign at least one variable to one, otherwise we may output all zeroes assignment.

4.3 Completeness

To finish the proof of Theorem 3 it remains to show a reduction in the other direction: from $K_n \setminus e$-free Edge Deletion to Min Horn Deletion. We achieve this goal by presenting an A-reduction from the $K_n \setminus e$-free Edge Deletion problem to another variant of MinOnes($\mathcal{F}$), which is Min Horn Deletion-complete.

▶ **Definition 22.** Let $n \ge 5$, and let $t = n(n-1)/2$. We define family of constraints $\mathcal{F}''_n = \{f_n, g_n\}$ as follows:
- $f_n(x_1, x_2, \ldots x_t) = 0$ if and only if exactly one of the variables takes value 1;
- $g_n(x_1, x_2, \ldots x_{t-1}) = 0$ if and only if all the variables take value 0.

The proof of the following lemma is a technical check that is essentially the same as the proof of Lemma 17. Hence, we leave it to the reader.

▶ **Lemma 23.** *For each $n \ge 5$, the set of constraints $\mathcal{F}''_n = \{f_n, g_n\}$ is weakly positive, and at the same time it is neither 0-valid, nor IHS-B for any B.*

Therefore, by Theorem 15 we know that MinOnes($\mathcal{F}''_n$) is Min Horn Deletion-complete and it suffices to present an A-reduction from $K_n \setminus e$-free Edge Deletion to MinOnes($\mathcal{F}''_n$).

▶ **Lemma 24.** *There is a polynomial-time algorithm, which given an instance G of $K_n \setminus e$-free Edge Deletion produces an instance $\mathcal{I}$ of MinOnes($\mathcal{F}''_n$), such that it is possible to remove exactly k edges in G to make it $K_n \setminus e$-free if and only if one can find a satisfying assignment for $\mathcal{I}$ that sets exactly k variables to 1.*

Proof. Consider an instance G of the $K_n \setminus e$-free Edge Deletion problem. We enumerate all the edges in the graph G as $e_1, e_2, \ldots, e_m$, and to each edge e_i we assign a fresh boolean variable x_i. For any induced subgraph H isomorphic to $K_n \setminus e$ we list all its edges $e_{i_1}, e_{i_2}, \ldots, e_{i_{t-1}}$ and create a corresponding constraint $g(x_{i_1}, x_{i_2}, \ldots, x_{i_{t-1}})$. For any induced clique K containing n vertices and edges $e_{i_1}, e_{i_2}, \ldots, e_{i_t}$, we create a constraint

$f(x_{i_1}, x_{i_2}, \ldots, x_{i_t})$. The output instance $\mathcal{I}$ of MinOnes($\mathcal{F}''_n$) is obtained by taking x_i to be the variable set, and putting all the constraints constructed above.

Note that if we delete some edges in the graph G, then an induced copy of the graph $K_n \setminus e$ can be obtained only on vertices that originally were inducing $K_n \setminus e$ or K_n. The constraints in the constructed instance guarantee that in each induced $K_n \setminus e$ subgraph at least one edge from the subgraph must be deleted, and in each induced subgraph K_n either at least two edges should be deleted, or none of the edges should be deleted. So, for any $S \subseteq \{1, 2, \ldots, |E(G)|\}$, the graph $G - F$, where $F = \{e_i : i \in S\}$, is $K_n \setminus e$-free if and only if the assignment $\{x_i = 1 \text{ iff } i \in S\}$ satisfies $\mathcal{I}$. This equivalence of solution sets immediately proves the lemma. ◀

As discussed earlier, Lemma 24 gives an A-reduction from $K_n \setminus e$-free Edge Deletion to MinOnes($\mathcal{F}''_n$), which is Min Horn Deletion-complete, thereby proving that $K_n \setminus e$-free Edge Deletion is A-reducible to Min Horn Deletion. This concludes the proof of Theorem 3.

5 Conclusions

In this work we initiated the study of approximability of edge modification problems related to the classes of H-free graphs. Mirroring known kernelization hardness results, we have shown that the problems are hard to approximate whenever H is a 3-connected graph with at least two non-edges, or it is a long enough path or cycle. It therefore seems that the approximation complexity of H-free Edge Deletion (Completion) somewhat matches the kernelization complexity in the cases considered so far, so it is tempting to formulate a conjecture that for every graph H, the H-free Edge Deletion (Completion) problem admits a polynomial kernel if and only if it admits a poly(OPT)-approximation algorithm. Since neither for kernelization nor for approximability the classification is close to being complete, this conjecture should be regarded as a very distant goal. However, one very concrete open question that arises is whether Cograph Edge Deletion (equivalent to $H = P_4$) admits a poly(OPT)-approximation. Here, we expect the answer to be positive, due to the existence of the polynomial kernel of Guillemot et al. [9]. The same question can be asked about the diamond graph, that is, a K_4 minus an edge; a polynomial kernel for Diamond-Free Edge Deletion was given by Cai [5]. Also, further investigation of the links between the case of a complete graph without one edge and the Min Horn Deletion problem, seems like an interesting direction.

References

1 N. R. Aravind, R. B. Sandeep, and Naveen Sivadasan. Parameterized lower bound and NP-completeness of some h-free edge deletion problems. In *COCOA 2015*, volume 9486 of *LNCS*, pages 424–438. Springer, 2015.

2 Ivan Bliznets, Marek Cygan, Paweł Komosa, Lukáš Mach, and Michał Pilipczuk. Lower bounds for the parameterized complexity of Minimum Fill-in and other completion problems. In *SODA 2016*, pages 1132–1151. SIAM, 2016.

3 Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996.

4 Leizhen Cai and Yufei Cai. Incompressibility of H-free edge modification problems. *Algorithmica*, 71(3):731–757, 2015.

5 Yufei Cai. Polynomial kernelisation of H-free edge modification problems. Master's thesis, The Chinese University of Hong Kong, 2012. Available at author's website.

6 Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, Erik Jan van Leeuwen, and Marcin Wrochna. Polynomial kernelization for removing induced claws and diamonds. *CoRR*, abs/1503.00704, 2015. To appear in the proceedings of WG 2015.

7 Pål Grønås Drange, Fedor V. Fomin, Michał Pilipczuk, and Yngve Villanger. Exploring the subexponential complexity of completion problems. *TOCT*, 7(4):14, 2015.

8 Archontia C. Giannopoulou, Daniel Lokshtanov, Saket Saurabh, and Ondrej Suchý. Tree deletion set has a polynomial kernel (but no $OPT^{O(1)}$ approximation). In *FSTTCS 2014*, volume 29 of *LIPIcs*, pages 85–96. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014.

9 Sylvain Guillemot, Frédéric Havet, Christophe Paul, and Anthony Perez. On the (non-)existence of polynomial kernels for P_ℓ-free edge modification problems. *Algorithmica*, 65(4):900–926, 2013.

10 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

11 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001. `doi:10.1137/S0097539799349948`.

12 Stefan Kratsch and Magnus Wahlström. Two edge modification problems without polynomial kernels. *Discrete Optimization*, 10(3):193–199, 2013.

13 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

14 Assaf Natanzon. Complexity and approximation of some graph modification problems. Master's thesis, Department of Computer Science, Tel Aviv University, 1999.

On Approximating Target Set Selection

Moses Charikar[1], Yonatan Naamad[2], and Anthony Wirth[3]

1 **Computer Science Department, Stanford University, Stanford, CA, USA**
`moses@cs.stanford.edu`
2 **Department of Computer Science, Princeton University, Princeton, NJ, USA**
`ynaamad@cs.princeton.edu`
3 **Department of Computing and Information Systems, The University of Melbourne, Parkville, Vic, Australia**
`awirth@unimelb.edu.au`

Abstract

We study the Target Set Selection (TSS) problem introduced by Kempe, Kleinberg, and Tardos (2003). This problem models the propagation of influence in a network, in a sequence of rounds. A set of nodes is made "active" initially. In each subsequent round, a vertex is activated if at least a certain number of its neighbors are (already) active. In the minimization version, the goal is to activate a small set of vertices initially – a seed, or target, set – so that activation spreads to the entire graph. In the absence of a sublinear-factor algorithm for the general version, we provide a (sublinear) approximation algorithm for the bounded-round version, where the goal is to activate all the vertices in r rounds. Assuming a known conjecture on the hardness of Planted Dense Subgraph, we establish hardness-of-approximation results for the bounded-round version. We show that they translate to general Target Set Selection, leading to a hardness factor of $n^{1/2-\varepsilon}$ for all $\varepsilon > 0$. This is the first polynomial hardness result for Target Set Selection, and the strongest conditional result known for a large class of monotone satisfiability problems. In the maximization version of TSS, the goal is to pick a target set of size k so as to maximize the number of nodes eventually active. We show an $n^{1-\varepsilon}$ hardness result for the undirected maximization version of the problem, thus establishing that the undirected case is as hard as the directed case. Finally, we demonstrate an SETH lower bound for the exact computation of the optimal seed set.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases target set selection, influence propagation, approximation algorithms, hardness of approximation, planted dense subgraph

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.4

1 Introduction

In this paper, we address the problem of targeting individuals to spread influence (or infection) in a network. Based on an average-case assumption about finding a planted dense subgraph, we develop the first polynomial-factor lower bound for a key minimization problem. Also, for a fixed-round version, we introduce the first sub-linear-factor approximation algorithm.

Motivated by work of Domingos and Richardson [14, 20], Kempe, Kleinberg, and Tardos [18] introduced the following model. A vertex is either active (infected) or inactive (uninfected). Given an initial *seed set* of active vertices, influence proceeds in a sequence of rounds. Every vertex v has a known, deterministic threshold $\tau(v)$. A previously inactive vertex v becomes active in a particular round if in the previous round at least $\tau(v)$ neighbors of v were active. Once a vertex is active, it remains active in all subsequent rounds. Since the process (essentially) stops if there is no new active vertex in some round, there are at most n rounds.

Both directed and undirected versions have been considered. In the directed case, the head vertex of an edge directly contributes to activating the tail vertex, but not vice versa.

The key question that arises in the study of viral marketing is TARGET SET SELECTION. Given the graph and the activation thresholds for vertices, which nodes should be initially targeted so as to spread the activation widely in the network? Specifically, we have these two problems,

- MIN-TSS: Find a minimum-size seed set that leads to all vertices eventually being active;
- MAX-TSS: Given budget k, find a size-k seed set that maximizes the final active set size.

Kempe, Kleinberg, and Tardos focused on the maximization problem where all thresholds are drawn randomly from a given range [18]. For the *directed* MAX-TSS problem, with deterministic thresholds, they showed that obtaining an $n^{1-\varepsilon}$ approximation is **NP**-hard. By reducing from LABEL COVER, Chen [7] showed that the minimization problem, MIN-TSS, cannot be approximated to within $2^{\log^{1-\varepsilon} n}$ unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\text{polylog}(n)})$, even for instances with uniform thresholds of 2. In fact [13], it is **NP**-hard to approximate LABEL COVER within $2^{\log^{1-\varepsilon} n}$, and this hardness bound extends to MIN-TSS.

Cicalese et al. [11, 10], considered versions of the problem in which the number of rounds is bounded. For graphs of bounded clique-width, given parameters α, β, and λ, they gave polynomial-time algorithms to determine whether there exists a target set of size β, such that at least α vertices are activated in at most λ rounds. Various other aspects of target set selection have been studied. For example, Coja-Oghlan et al. [12] obtained combinatorial bounds for the size of target sets in expanders, while Ben-Zwi et al. [5] obtained upper and lower bounds for this problem on low-treewidth graphs.

Our Results

We seek a polynomial-factor lower-bound for approximating MIN-TSS. Obtaining such a result by reduction from known **NP**-hard problems would be a breakthrough. As we point out in Section 7, (the bounded-round version of) MIN-TSS belongs to the *MMSA hierarchy* [13]. A prevailing definition of this class of problems is: Given a monotone Boolean circuit, minimize the number of inputs set to `True` so that the circuit evaluates to `True`. No problem in the MMSA hierarchy is currently known to have a polynomial hardness result.[1] We derive a polynomial-factor hardness result for MIN-TSS from an average-case hardness conjecture for the PLANTED DENSE SUBGRAPH problem (see Section 3.2)

In Section 4.2, we show the following hardness result for the bounded-round version of MIN-TSS. Assuming the PLANTED DENSE SUBGRAPH CONJECTURE, for every $\varepsilon > 0$, r-round MIN-TSS is $n^{1/2-1/2^{\lfloor r/2 \rfloor}-\varepsilon}$ hard to approximate. In Section 4.3, we prove the corollary that, assuming the PLANTED DENSE SUBGRAPH CONJECTURE, MIN-TSS (with unbounded number of rounds) is **NP**-hard to approximate within $n^{1/2-\varepsilon}$ for every $\varepsilon > 0$.

After this, in Section 5, we provide an $O((\tau_{\max}/\tau_{\min})^{1-1/r} n^{1-1/r} \log^{1/r} n)$ approximation algorithm for r-round MIN-TSS, where $\tau_{\max}$ and $\tau_{\min}$ are the maximum and minimum thresholds in the instance, respectively. Subsequently, we show in Section 6 that *undirected* MAX-TSS is as hard as the *directed* version, giving an $n^{1-\varepsilon}$ hardness for undirected MAX-TSS. Finally, in Section 8, we reuse ideas from Section 4.3 to give an $O(n^k)$ SETH-lower

[1] As explained in Section 7, another variant of MMSA, where the circuits may use non-monotone gates in computing a monotone function, is known to admit problems that are $O(n^{1-\varepsilon})$ hard to approximate unless $\mathbf{NP} \subseteq \mathbf{DTIME}[n^{O(\log n)}]$ [21].

bound on the time needed to find contagious sets of size k, for $k = O(1)$, which is tight up to near-linear factors.

2 Outline of Key Technical Ideas

The main goal of our paper is to obtain a better understanding of the complexity of MIN-TSS. We focus in particular on the bounded-round variant, which has previously only been studied from the point of view of fixed-parameter tractability. As we show, understanding the hardness of the bounded-round variant is a good stepping stone in obtaining hardness results for (unbounded) MIN-TSS.

2.1 Hardness of Min-TSS

A $2^{\log^{1-\varepsilon} n}$ hardness factor is known for MIN-TSS, but there are no non-trivial approximation upper bounds for this problem. We obtain a polynomial hardness factor based on the planted dense subgraph conjecture by introducing and exploiting a recursive version. Roughly speaking, the planted dense subgraph conjecture says that it is hard to distinguish random graphs of degree n^α from such graphs where $k = \sqrt{n}$ vertices have a planted dense subgraph of degree k^β, for $\beta < \alpha$.

To assist our exposition of the intuition, we will sacrifice a little accuracy, and think of $\beta = \alpha = 1/2$. So (pretend that) it is hard to distinguish random graphs of degree $\sqrt{n}$ (unplanted) from such graphs where a subset of $\sqrt{n}$ vertices have a planted random graph of degree $n^{1/4}$ (planted) – a factor of roughly $n^{1/4}$ larger than the expected vertex degree in a random subgraph of that size. Consider these graph families as inputs to MIN-TSS with the threshold set to a large constant. In the unplanted case, we can show that at least $n^{1/2}$ vertices must be initially activated in order to activate all vertices in a constant number of rounds. On the other hand, in the planted case, a target set of roughly $n^{1/4}$ vertices will activate all vertices in at most four rounds: in two rounds, all the vertices in the planted set will be activated, and in two more rounds all vertices in the graph will be activated. This leads to an $\Omega(n^{1/4})$ hardness result for 4-round MIN-TSS.

In order to show stronger hardness results (for more rounds), we recurse. Consider a recursively planted dense subgraph instance where we start with a random graph of degree $n^{1/2}$ and plant a random subgraph of $n^{1/2}$ vertices and degree $n^{1/4}$. Within this, we plant a random subgraph of $n^{1/4}$ vertices and degree $n^{1/8}$, and so on; the last subgraph in this sequence is on $n^{1/2^t}$ vertices with degree $n^{1/2^{t+1}}$. (Again, to facilitate the explanation, the parameters are slightly inaccurate.) We show that the planted dense subgraph conjecture implies the hardness of the recursive version. As before, in the unplanted case, at least $n^{1/2}$ vertices must be initially activated in order to activate all vertices in a constant number of rounds. In the recursively planted case, activating roughly $n^{1/2^{t+1}}$ vertices will activate all vertices in at most $2(t+1)$ rounds: in two rounds, all the vertices of the inner-most planted subgraph will be activated, in a further two rounds, all vertices of the second-deepest planted subgraph will be activated, and so on. This establishes a hardness of $n^{1/2-1/2^{t+1}}$ for $2(t+1)$-round MIN-TSS.

Via a direct reduction, we show that the hardness results for r-round MIN-TSS imply hardness for (unbounded-round) MIN-TSS. For every constant $\varepsilon > 0$, we show that MIN-TSS is $n^{1/2-\varepsilon}$ hard to approximate. The reduction is easiest to describe with directed edges, but these can be simulated with a gadget comprising undirected edges. Given an instance of r-round MIN-TSS on $G(V,E)$, the construction consists of $2r+1$ layers of copies of vertices of V: layers $S_0, S_1, \ldots, S_r$ are interleaved with layers $M_0, \ldots, M_{r-1}$. Layer S_0 contains the seed set and for each $i > 0$, vertices active in S_i represent the set of vertices active in G after i rounds. Each layer M_i contains "memory" vertices: the copy of vertex v in this layer is activated if a copy of v is activated in at least one of the previous layers, $S_0, \ldots, S_i$. Between layers M_i and S_{i+1} we place a bipartite copy of the original graph G, and the copy of v in S_{i+1} has the same threshold as v in the original graph. This simulates one round of the activation process on the original graph. Finally, we place a complete bipartite graph directed from vertices in S_r to those in S_0: each vertex in S_0 has threshold $|V|$, so, unless in the seed set, it is activated if and only if all vertices in S_r are active.

The recursive planted dense subgraph construction and its application are new. It would be interesting to understand what sequences of parameters in the recursive construction imply indistinguishability between the unplanted case and the recursively planted case. Our hybrid-like argument for indistinguishibility relies on the fact that the recursively planted instances are scaled-down copies of hard instances of planted dense subgraph; however, this might not be needed in the recursive construction. Together with a better understanding of planted dense subgraph in regimes where the planted subgraph size $k \gg \sqrt{n}$, this could lead to tight hardness results for MIN-TSS. Although we are unaware of an algorithm for MIN-TSS with approximation factor $o(n)$, our current construction establishes hardness at most $n^{1/2}$ because the unplanted case has $\text{OPT} = O(n^{1/2})$. Establishing stronger hardness results will need constructions where OPT is much higher; this could be achieved by setting the order of the planted subgraph to be $n^{1-\varepsilon}$, with appropriate choices of degrees for the planted and ambient graphs. Because the assumptions were originally formulated to capture the hard case for approximating densest subgraph, the stated assumptions in the literature about hardness of planted dense subgraph only apply to $k \leq \sqrt{n}$. A more comprehensive understanding of planted dense subgraph in the $k > \sqrt{n}$ regime would be interesting in its own right, and could lead to an almost-tight $n^{1-\varepsilon}$ hardness for MIN-TSS.

2.2 Approximation of r-round Min-TSS

Until now, no non-trivial approximation algorithms for bounded-latency MIN-TSS is known. When all thresholds are the same, our algorithm follows a greedy approach and obtains an $\tilde{O}(n^{1-1/r})$-approximation for r-round MIN-TSS. In general, the approximation factor also depends on the ratio of the largest to smallest thresholds. The challenge in applying a greedy approach to MIN-TSS is quantifying the progress made in adding a single vertex to the seed set. Indeed, a single vertex may have negligible impact until several other vertices are picked. The key idea behind our algorithm is a potential function, called *hunger*, that guides the algorithm. Given a seed set S and a bound r on the number of rounds, a vertex v has positive hunger if and only if it remains inactive after r rounds. In this case, v's hunger is the number of additional neighbors that need to be active after $r - 1$ rounds, in order to activate v in the next round.

Our algorithm chooses the seed set in two phases, based on a parameter β that we choose appropriately to obtain the approximation guarantee. In the first phase, we greedily pick vertices that have more than β neighbors that would not otherwise become active within one round. The first phase ensures that in the "residual" graph, degrees of vertices are bounded by β. With this, we can relate the size of the seed set picked in the second phase to the optimum seed set size. In the second phase, given the current seed set, we repeatedly pick vertices greedily to reduce the total hunger. Our analysis shows that one of the vertices of the optimal solution reduces the total hunger by a significant quantity. However, we lose a factor of β^{r-1} in relating the drop in the total hunger to the effect of this vertex on the optimal solution. Consequently, in the second phase, the bound on the size of the picked seed set is (roughly) a factor β^{r-1} times optimal.

3 Preliminaries

3.1 Formal definition and notation

An instance of Target Set Selection (TSS) is an n-vertex (di)graph $G = (V, E)$ coupled with a threshold function $\tau : V \to \mathbb{Z}^+$. Seed set $S \subset V$ comprises the vertices that are *active in round* 0. For all $t > 0$, a vertex $v \in V$ is called *active in round* t if either it is active in round $t-1$ or at least $\tau(v)$ of its (in)neighbors were active in round $t-1$. The r^{th}*-round activation family* of a seed set $S \subset V$, denoted by $\mathcal{A}_r(S)$, comprises those vertices active in round $t = r$, conditioned on exactly the vertices in S being active at time $t = 0$. The *activation family* of seed set S is $\mathcal{A}_\infty(S) \equiv \lim_{r\to\infty} \mathcal{A}_r(S)$. By monotonicity, and the Markovian nature of this process, it is easy to verify that $\mathcal{A}_\infty(S)$ is equivalent to $\mathcal{A}_{n-1}(S)$. We study three variants of the Target Set Selection problem.

TSS: Given G and τ, what is the size of the smallest seed set S for which $\mathcal{A}_\infty(S) = V$?
RTSS$_r$: Given G and τ, what is the size of the smallest seed set S for which $\mathcal{A}_r(S) = V$?
MaxTSS: Given G, τ, and $k > 0$, conditioned on $|S| \leq k$, what is the largest value of $|\mathcal{A}_\infty(S)|$?

We sometimes refer to TSS as Min-TSS, and RTSS stands for r-Round TSS. For both minimization problems, a seed set whose activation family (within the round limit, if any) is the whole graph is called a *contagious* set. When all thresholds are equal, we may abuse notation and let τ itself be an integer. All tuples in this paper are written thus $\vec{a}$, and are indexed in two ways. If $\vec{\xi} = (\xi_1, \xi_2, \cdots, \xi_m)$ is a tuple and $1 \leq i < j \leq m$, we let $\vec{\xi}_i^j$ denote the contiguous sub-tuple $(\xi_i, \xi_{i+1}, \cdots, \xi_j)$. Sometimes, we are interested in the suffix of the tuple, so we can index the final elements thus: $\vec{\xi}_{(-i)}^{(-1)} = (x_{m-(i-1)}, x_{m-(i-2)}, \ldots, x_m)$.

3.2 Planted Dense Subgraph Conjecture

The Planted Dense Subgraph (PDS) problem is a generalization of Planted Clique, in which the goal is to distinguish a $G(n, p)$, Erdős-Rényi, random graph from one that contains a planted dense Erdős-Rényi component. Formally, an instance of the problem PDS(n, k, α, β) is parameterized by the graph order n, the subgraph order k, and log-densities $\alpha, \beta \in (0, 1)$. We are then asked for an algorithm that with high probability distinguishes between these two families of random graphs:

Unplanted: An Erdős-Rényi random graph $G(n, n^{\alpha-1})$ (i.e., a random graph with expected degree $\approx n^\alpha$).
Planted: An Erdős-Rényi random graph $G(n, n^{\alpha-1})$ from which k vertices are chosen uniformly at random and their induced subgraph is replaced by an instance of $G(k, k^{\beta-1})$.

The input to the PDS(n, k, α, β) problem is a graph, with the *promise* that with probability 1/2 it is drawn from the Planted distribution and with probability 1/2 it is drawn from the Unplanted distribution. The output is True or False, indicating whether the graph has a planted subcomponent. An algorithm *solves* the problem if, for some $\varepsilon > 0$, independent of n, it attains an ε advantage over random guessing. That is, with probability at least $1/2 + \varepsilon$, it correctly states from which of the two distributions the input graph was drawn. This statement about probability is over the joint distribution of the input graph and the random choice sequence of the algorithm.

As observed by Bhaskara et al. [6], PDS(n, k, α, β) admits a simple polynomial-time deterministic algorithm when $\beta > \alpha$; for $k > \sqrt{n}$, an eigenvalue approach works with a weaker condition on β. For $\beta < \alpha$ and $k \leq \sqrt{n}$, there is no quasipolynomial-time algorithm (deterministic or randomized) known for PDS, giving rise to the Planted Dense Subgraph Conjecture, viz.

▶ **Conjecture 1.** *For every $\varepsilon > 0$, $k \leq \sqrt{n}$, and $\beta < \alpha$, no probabilistic polynomial-time (PPT) algorithm can, with advantage greater than ε, solve* PDS(n, k, α, β)*.*

This conjecture is attractive because the current best-known algorithms for the problem require $2^{n^{\Theta(1)}}$ time (as opposed to $n^{\Theta(\log n)}$ for the well-studied Planted Clique problem).

Similar conjectures have previously been made in different contexts in theoretical computer science [2, 3, 4]. The precise form of the conjecture we state is very similar to the conjecture stated by Awasthi et al. [4]. As we show in Section 4, Planted Dense Subgraph also naturally lends itself to showing hardness for the bounded-round version of Min-TSS, which in turn leads to a hardness result for the (unbounded) Min-TSS problem.

4 Hardness of Min-TSS

In this section, we prove that the Planted Dense Subgraph (PDS) conjecture implies that for all $\varepsilon > 0$, there is no probabilistic polynomial-time algorithm for Min-TSS that achieves an approximation factor of $O(n^{1/2-\varepsilon})$. We first show that the Planted Dense Subgraph conjecture implies the hardness of a recursive version, and we use this recursive version to show hardness for Min-TSS.

4.1 Recursive extension of PDS

To simplify notation, we define a right-associative operator, $\triangleleft$, on distributions of graphs. Suppose there are two distributions on graphs, $\mathcal{G}_1$, on graphs of order n, and $\mathcal{G}_2$, on graphs of order n', with $n > n'$. The distribution $\mathcal{G}_1 \triangleleft \mathcal{G}_2$ is defined thus:

Draw a graph G_1 from $\mathcal{G}_1$ and a graph G_2 from $\mathcal{G}_2$, then choose uniformly at random a subset S' of vertices of size n' from G_1 and *replace* its induced subgraph with G_2.

Hence PDS(n, k, α, β) asks us to state whether a graph is drawn from $G(n, n^{\alpha-1}) \triangleleft G(k, k^{\beta-1})$ (True) or from $G(n, n^{\alpha-1})$ (False).

In the definition of PDS(n, k, α, β), one consequence of the random construction of both $G(n, n^{\alpha-1})$ and $G(k, k^{\beta-1})$ is that the planting process can be naturally recursed. For every pair of length-m tuples $\vec{n} = (n_1, n_2, \ldots, n_m)$ – the subgraph orders – and $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ – the subgraph log-densities – with, for each i, $n_i \in \mathbb{Z}^+$, $n_i > n_{i+1}$, and $\alpha_i \in (0, 1)$, we define the PDS$^m(\vec{n}, \vec{\alpha})$ distribution via the recurrence

$$\mathrm{PDS}^m(\vec{n}, \vec{\alpha}) = \begin{cases} G(n_1, {n_1}^{\alpha_1 - 1}) \triangleleft \mathrm{PDS}^{m-1}(\vec{n}^{(-1)}_{(-m+1)}, \vec{\alpha}^{(-1)}_{(-m+1)}) & \text{if } m > 1; \\ G(n_1, {n_1}^{\alpha_1 - 1}), & \text{otherwise.} \end{cases}$$

We also define the (eponymous) PDS$^m(\vec{n}, \vec{\alpha})$ problem: distinguish with ε advantage graphs drawn from the PDS$^m(\vec{n}, \vec{\alpha})$ distribution from those drawn simply from PDS$^1(n_1, \alpha_1) = G(n_1, n^{\alpha_1 - 1})$. More formally, under the promise that with probability 1/2 the graph is from the former distribution, and with probability 1/2 from the latter, an algorithm solves the problem if with probability at least $1/2 + \varepsilon$ it correctly states which of the distributions the graph came from. Setting $m = 2$, $\vec{n} = (n, k)$, and $\vec{\alpha} = (\alpha, \beta)$ recovers exactly PDS(n, k, α, β).

We now show that for monotonically decreasing log-densities, recursive planting of small, but polynomially sized, subgraphs leads to a problem no simpler than PDS.

▶ **Lemma 2.** *Assuming Conjecture 1, if $m \geq 2$ is a constant, $\alpha_i > \alpha_{i+1}$ for every $i < m$, and for some constant $c > 0$, $n_{i+1} \in [n_i^c, \sqrt{n_i}]$, then no PPT algorithm can solve the* $\mathrm{PDS}^m(\vec{n}, \vec{\alpha})$ *problem with ε advantage.*

Proof (by contradiction). Consider the *minimum* m for which some algorithm A solves $\mathrm{PDS}^m(\vec{n}, \vec{\alpha})$ with ε advantage. We show how to construct an algorithm that attains an $\varepsilon' > 0$ advantage for the problem $\mathrm{PDS}^{m-1}(\vec{n}^{(-1)}_{(-m+1)}, \vec{\alpha}^{(-1)}_{(-m+1)})$, contradicting the minimality of m. For a distribution $\mathcal{H}$, let $\mathcal{F}(\mathcal{H})$ be the distribution $G(n_1, n_1^{\alpha_1 - 1}) \triangleleft \mathcal{H}$. Hence

$$\mathcal{F}\left(\mathrm{PDS}^{m-1}(\vec{n}^{(-1)}_{(-m+1)}, \vec{\alpha}^{(-1)}_{(-m+1)})\right)$$

is $\mathrm{PDS}^m(\vec{n}, \vec{\alpha})$ and $\mathcal{F}\left(G(n_2, {n_2}^{\alpha_2 - 1})\right)$ is $\mathrm{PDS}^2\left((n_1, n_2), (\alpha_1, \alpha_2)\right)$. But, of course, this $\mathcal{F}$ operator represents a (randomized) polynomial-time-computable operation on a graph, once drawn from distribution $\mathcal{H}$. Assuming the existence of algorithm A, we propose algorithm $A_{\mathcal{F}}$ for $\mathrm{PDS}^{m-1}(\vec{n}^{(-1)}_{(-m+1)}, \vec{\alpha}^{(-1)}_{(-m+1)})$:

Given graph H, apply algorithm A to $\mathcal{F}(H)$ and return A's answer.

For the following three distributions, let p_i be the probability that A reports TRUE when the graph is drawn from distribution i.

1. $G(n_1, {n_1}^{\alpha_1 - 1})$;
2. $\mathcal{F}\left(G(n_2, {n_2}^{\alpha_2 - 1})\right) = \mathrm{PDS}^2\left((n_1, n_2), (\alpha_1, \alpha_2)\right)$;
3. $\mathcal{F}\left(\mathrm{PDS}^{m-1}(\vec{n}^{(-1)}_{(-m+1)}, \vec{\alpha}^{(-1)}_{(-m+1)})\right) = \mathrm{PDS}^m(\vec{n}, \vec{\alpha})$

Since we assumed the PLANTED DENSE SUBGRAPH CONJECTURE, the probability of returning the correct answer when distinguishing between distributions 1 and 2 is at most $1/2 + o(1)$. That is, $(1 - p_1)/2 + p_2/2 \leq 1/2 + o(1)$. On the other hand, since A solves $\mathrm{PDS}^m(\vec{n}, \vec{\alpha})$, with ε advantage, $(1 - p_1)/2 + p_3/2 \geq 1/2 + \varepsilon$.

Consider algorithm $A_{\mathcal{F}}$, it applies $\mathcal{F}$ to its input graph and sends it to A. Although A was promised a graph that was with probability $1/2$ from the first distribution and with probability $1/2$ from the third[2], its input is merely a graph with nonzero mass in each of the two distributions, and thus it outputs some value in probabilistic polynomial time[3]. The probability of algorithm $A_{\mathcal{F}}$ correctly reporting TRUE is (by definition) $(1 - p_2)/2 + p_3/2$, which, from the two previous inequalities, is easily seen to be at least $1/2 + \varepsilon - o(1)$. Hence, for sufficiently large n, Algorithm $A_{\mathcal{F}}$ is a PPT algorithm with advantage $\varepsilon/2$ for the $\mathrm{PDS}^{m-1}(\vec{n}^{(-1)}_{(-m+1)}, \vec{\alpha}^{(-1)}_{(-m+1)})$ problem, contradicting the minimality of m. ◀

4.2 Hardness of fixed-round TSS

We now prove the following theorem on the hardness of r-round MIN-TSS.

[2] This application of a promise oracle to non-promise-satisfying inputs is an example of a *non-smart* reduction [16].

[3] Although the positivity of the mass on some order-n graph might seem like a technicality, it is not intrinsic to the analysis. For other, similar, distributions of graphs where $\mathcal{F}(H)$ may have 0 probability in either distribution, we can run A up to its polynomial-time upper-bound for promise-satisfying instances and return FALSE if it has failed to return by then. The rest of the analysis proceeds identically, up to the necessary modifications needed to handle the new ensemble of distributions.

▶ **Theorem 3.** *Assuming the* Planted Dense Subgraph Conjecture, *for every $\varepsilon > 0$, no PPT algorithm can approximate r-round* Min-TSS *to within a factor of $O(n^{1/2-1/2^{\lfloor r/2 \rfloor}-\varepsilon})$.*

The general idea is as follows. We first prove (in Lemma 4) a lower bound on the size of r-round contagious sets on $G(n_1, {n_1}^{\alpha_1 - 1})$ for particular (ranges of) values of n_1 and α_1. Afterwards, we argue (in Lemma 5) that for some appropriate setting of $\vec{n} = (n_1, n_2, \cdots, n_{\lfloor r/2 \rfloor})$ and $\vec{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_{\lfloor r/2 \rfloor})$, the $n_{\lfloor r/2 \rfloor}$ vertices in the densest component of $\mathrm{PDS}^{\lfloor r/2 \rfloor}(\vec{n}, \vec{\alpha})$ must form a contagious set of size $n_{\lfloor r/2 \rfloor}$. Theorem 3 then follows from a direct application of Lemma 2.

▶ **Lemma 4.** *For all $\alpha \in (0, 1/2]$ and all positive integers r, τ, the r-round* Min-TSS *instance* $\mathrm{RTSS}_r(G(n, n^{\alpha-1}), \tau)$ *has $|OPT| = \tilde{\Omega}(n^\beta)$, where $\beta = (\tau(1-\alpha) - 1)/(\tau - 1)$.*

Proof. For a seed set S of size $\tilde{O}(n^\beta)$, chosen uniformly at random (u.a.r.), and a vertex $v \notin S$, chosen u.a.r., the probability that v has k neighbors in S is, by the union bound, at most $\binom{|S|}{k}(n^{\alpha-1})^k = \tilde{O}\left((n^\beta n^{\alpha-1})^k\right)$ (for constant k), which decreases geometrically with k. Thus, the number of vertices newly activated after one round, $|\mathcal{A}_1(S) \setminus S|$, follows a binomial distribution with mean $\mu = \tilde{O}(n(n^\beta n^{\alpha-1})^\tau) = \tilde{O}(n^{1+(\alpha+\beta-1)\tau})$. Because $(\alpha+\beta-1)\tau = \beta - 1$, this expression simplifies to $\tilde{O}(n^\beta)$. Chernoff bounds tell us that the probability that $|\mathcal{A}_1(S) \setminus S|$ exceeds its expectation by a $\log^2 n$ factor is bounded by

$$\exp[\mu(\log^2 n - 1 - 2\log^2 n(\log\log n))] < \exp[\mu(-\log^2 n)] = n^{-\mu \log n}\,.$$

Given it has size $\tilde{O}(n^\beta)$, there are $\tilde{O}\left(\binom{n}{n^\beta}\right)$ choices of seed set S. By the union bound, the probability that at least one of them activates more than $\tilde{O}(n^\beta)$ vertices within one round is $1/n^{\Omega(1)}$. By induction, after a constant number of activation rounds, for all $\varepsilon > 0$, the number of active nodes is with high probability at most $o(n^{\beta+\varepsilon})$. Hence, with high probability, no seed set of size $\tilde{O}(n^\beta)$ is an r-round contagious set. ◀

▶ **Lemma 5.** *Suppose, in an instance of the r-round* Min-TSS *problem, every vertex shares the same constant threshold τ, with*

$$\rho \equiv \lfloor r/2 \rfloor, \qquad \varepsilon_0 \in \left(0, \frac{1}{3(\tau+1)^2}\right), \qquad \vec{n} = \left(n, \sqrt{n}, \sqrt{\sqrt{n}}, \ldots, \sqrt[2^\rho]{n}\right), \textit{ and}$$

$$\vec{\alpha} = \left(\frac{1}{2}, \frac{1}{2} - \frac{\varepsilon_0}{1+\rho}, \frac{1}{2} - \frac{2\varepsilon_0}{1+\rho}, \ldots, \frac{1}{2} - \frac{\rho\varepsilon_0}{1+\rho}\right).$$

If $G \sim \mathrm{PDS}^\rho(\vec{n}, \vec{\alpha})$, then $\mathrm{RTSS}_r(G, \tau)$ has $|OPT| = O(\sqrt[2^\rho]{n})$.

Proof. For $i \in 1, 2, \cdots, 1+\rho$, let G_i be the depth-i planted component of G, i.e., the graph of order $k_i = \sqrt[2^{i-1}]{n}$ and average degree ${k_i}^{1/2-(i-1)\varepsilon_0/(1+\rho)}$ planted in G. Further, for every i, define $a_i = {k_i}^{-1/2-\varepsilon_0}$, making a_i a lower bound on the edge probability in G_i. For $i \in \{1, 2, \ldots, \rho\}$, choosing a subset of size $\sqrt{k_i} = k_{i+1}$ of G_i's vertices u.a.r. to be the seed set S, leads to random variable $|\mathcal{A}_1(S) \setminus S|$ following a binomial distribution with $\mu = \Omega(k_i \cdot (\sqrt{k_i} a_i)^\tau)$. Each indicator of membership in $\mathcal{A}_1(S) \setminus S$ is a Bernoulli random variable with probability (of activation) at least $\binom{\sqrt{k_i}}{\tau}{a_i}^\tau$. The expression for the mean is thus $\Omega(k_i(\sqrt{k_i} \cdot {k_i}^{-1/2-\varepsilon_0})^\tau)$, which is $\Omega({k_i}^{1-\tau\varepsilon_0})$. The probability that, for a randomly chosen seed set of size $\sqrt{k_i}$, fewer than $\mu/2$ vertices become active is, again by the Chernoff bounds, less than $e^{-\Omega(\mu)} = n^{-\Omega(\mu/\log n)} = n^{-\Omega({k_i}^{1-\tau\varepsilon_0}/\log n)} < n^{-k_i^{2/3}} \ll n^{-\sqrt{k_i}-2}$. Every remaining inactive vertex in G_i would now have $\hat{d} = \mu a_i = \tilde{\Omega}({k_i}^{1/2-\varepsilon_0(\tau+1)}) = \tilde{\Omega}({k_i}^{1/2-1/(3(\tau+1))})$ activated neighbors in expectation. Thus, with high probability, no vertex has fewer than $\tilde{\Omega}({k_i}^{1/3})$ active neighbors. Since each vertex has so many active neighbors, with high

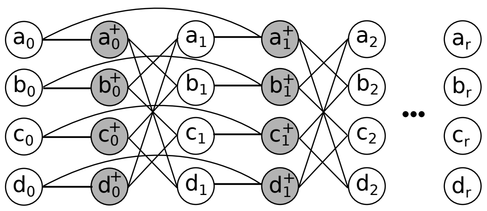


Figure 1 Sample of the first few layers of the r-round Min-TSS to TSS reduction. The i-th stage vertices are white and the memory vertices are grey. All drawn arcs are oriented rightwards. Not shown: directed edges from each vertex in S_r to all those in S_0.

probability, all of G_i's vertices becomes active within two rounds of all of G_{i+1}'s vertices becoming active. By induction, activating all of $G_{r'+1}$'s vertices will with high probability activate all of G_1's vertices after $2\rho \leq r$ rounds. Since $|V(G_{\rho+1})| = n^{1/2^\rho}$, the lemma follows. ◀

Proof of Theorem 3. Given an instance G of the $\text{PDS}^m(\vec{n}, \vec{\alpha})$ problem with $\vec{n}$ and $\vec{\alpha}$ satisfying the conditions of Lemma 5 (and thus Lemma 2), we choose a threshold τ satisfying $(\tau/2-1)/(\tau-1) > 1/2 - \varepsilon/2$ and generate the r-round Min-TSS instance $\text{RTSS}_r(G, \tau)$. If G comes from the unplanted distribution, an application of Lemma 4 provides a lower bound of $\tilde{\Omega}(n^{1/2-\varepsilon/2})$ on the size of some optimal seed set, OPT. On the other hand, if G comes from the planted distribution, Lemma 5 provides an upper bound of $O(n^{1/2^{\lfloor r/2 \rfloor}})$ on the size of OPT. Thus a PPT algorithm with approximation factor in

$$\tilde{O}(n^{1/2-\varepsilon/2}/n^{1/2^{\lfloor r/2 \rfloor}}) = \tilde{O}(n^{1/2-1/2^{\lfloor r/2 \rfloor}-\varepsilon/2}) = O(n^{1/2-1/2^{\lfloor r/2 \rfloor}-\varepsilon})$$

for r-round Min-TSS can distinguish the two cases, contradicting the Planted Dense Subgraph Conjecture. ◀

4.3 Hardness of round-unbounded TSS

We now show that for every constant r the general form of Min-TSS is, up to a constant factor, at least as hard to approximate as r-round Min-TSS.

▶ **Theorem 6.** *An $O(f(n))$-approximation algorithm for* Min-TSS *can be transformed into an $O(f(n))$-approximation algorithm for r-round* Min-TSS.

Proof. Our reduction relies heavily on directed edges. The hardness for undirected Min-TSS follows by simulating each directed edge with a *directed-edge gadget*, as shown in the left part of Figure 2. Given an instance $\text{RTSS}_r(G = (V, E), \tau)$ of r-round Min-TSS, we create an instance $\text{TSS}(G', \tau')$ of TSS as follows, and as depicted in Figure 1.

1. For $i = 0, 1, \ldots, r$, and for each $v \in V$, there is a vertex v_i. Collectively, the vertices $\{v_i\}_{v \in V}$ constitute S_i, the "i-th stage" vertices of G'.
2. For $j = 0, 1, \ldots, r-1$, and for each $v \in V$, there is a vertex v_j^+. Collectively, the vertices $\{v_j^+\}_{v \in V}$ constitute M_j, the "j-th stage memory vertices" of G'.
3. For $i = 0, 1, \ldots, r-1$, and for $j = i, \ldots, r-1$, there is a directed edge (gadget) from v_i to v_j^+.
4. For $i = 0, 1, \ldots, r-1$, and for each pair (u, v) in E, there is a directed edge (gadget) from u_i^+ to v_{i+1}.
5. For each vertex $x \in S_r$ and for each $y \in S_0$ there is a directed-edge (gadget) from x to y.

6. For each $v \in V$, and $i = 0, \ldots, r$ and $j = 0, \ldots, r-1$, $\tau'(v_0) = |V|$, $\tau'(v_i) = \tau(v)$, and $\tau'(v_j^+) = 1$.

The goal of the construction is for the active vertices in S_0 to represent (at least initially) the seed set S and for the active vertices in S_i to represent $\mathcal{A}_i(S)$ in G. If S_r is entirely active – representing $\mathcal{A}_r(S) = V$ – this set in turn activates all of S_0, causing all vertices in G' to become active. With the exception of those from S_r to S_0, the *directed* edges ensure that vertices in some stage cannot activate those in an "earlier" stage.

In designing G', simply linking vertices in S_i to those in S_{i+1} with directed edges is inappropriate. We need to ensure that a vertex in G that is active in round i, represented by an active vertex in stage i of G', is (also) represented as active in stage $i+1$ of G'. That is, if all v_i are active for $v \in \mathcal{A}_i(S)$, then all v_{i+1} are also active for $v \in \mathcal{A}_i(S)$. To assist us in this, the memory vertex v_j^+, in M_j is active whenever there is some v_i, with $i < j$, that is active.

Thus, a contagious set for the RTSS instance corresponds exactly to a contagious set in S_0. To show that G' does not contain smaller contagious sets, observe that activating v_0 weakly dominates activating each of v_i, for $i > 0$, and v_i^+, for $i \geq 0$. Vertex v_i will become active within i rounds anyway, so there is no benefit in activating it earlier; a similar argument applies to v_i^+. Thus, a contagious set containing vertices in $V(G') \setminus S_0$ can be transformed into a no-larger contagious set entirely inside S_0. Hence the optimal values of both this instance G' of TSS and of the initial instance G of RTSS are equal. ◀

Picking r sufficiently large, Theorem 3 and Theorem 6 together imply the following theorem.

▶ **Theorem 7.** *Assuming the* Planted Dense Subgraph Conjecture, *for no $\varepsilon > 0$ can a PPT algorithm approximate* Min-TSS *to within a factor of $O(n^{1/2-\varepsilon})$.*

5 Approximation of r-round Min-TSS

In Section 4.2, we showed that the Planted Dense Subgraph Conjecture implies $O(n^{1/2-1/2^{\lfloor r/2 \rfloor}-\varepsilon})$ hardness for the r-round Min-TSS problem, even in instances where all vertices have thresholds bounded by a constant. In this section, we complement the hardness result with an $\tilde{O}(n^{1-1/r})$ approximation algorithm for such graphs, viz.

▶ **Theorem 8.** *For every $r \in \mathbb{Z}^+$, there is a polynomial-time algorithm approximating r-round TSS to within a factor of $O((\tau_{\max}/\tau_{\min})^{1-1/r} n^{1-1/r} \log^{1/r} n)$.*

Overall, the algorithm follows a two-step process. A *degree-reduction* step – where we greedily add vertices to the seed set S whose neighborhoods contain many vertices that would not be in $\mathcal{A}_1(S)$ for the S thus far – followed by a *greedy* step – where we greedily select vertices that most reduce a potential function we call the *total hunger*. Roughly, if the seed set S would activate vertex v in r rounds, then v's hunger is zero; otherwise, its is the number of active neighbors v still "lacks" at the end of the $(r-1)^{\text{th}}$ round. The total hunger of the graph is the sum of these vertex hungers. We then argue that each vertex chosen in the degree-reduction step necessarily reduces the total hunger by a large amount. After these degree-reducing vertices have been added to the seed set, the residual graph has bounded degree, which means that the greedy algorithm is effective.

Formally, for a vertex v and seed set S, define v's *hunger* thus, where $\delta(v)$ is the set of v's neighbors:

$$h_{S,r}(v) = \begin{cases} 0, & \text{if } v \in \mathcal{A}_r(S) \\ \tau(v) - |\{u : u \in \delta(v), u \in \mathcal{A}_{r-1}(S)\}|, & \text{otherwise.} \end{cases}$$

Algorithm 1 Algorithm for computing a target set: parameter β specified below.

```
1: $S_1 \leftarrow \varnothing, S_2 \leftarrow \varnothing$  {Initialization}
2: while There exists some vertex $u$ with more than $\beta$ $(S_1, 1)$-hungry neighbors do
3:     Add $u$ to $S_1$. {Degree-reduction step}
4: end while
5: while There exists an $(S_1 \cup S_2, r)$-hungry vertex do
6:     Add to $S_2$ the vertex that most reduces the total hunger of $(S_1 \cup S_2, r)$-hungry vertices.
       {Greedy step}
7: end while
8: Return $S_1 \cup S_2$.
```

Likewise, define the total (S, r) hunger in the graph thus, $h_r(S) = \sum_{v \in V} h_{S,r}(v)$. Vertex v is called (S, r)-hungry if $h_{S,r}(v) > 0$. Algorithm 1, including a parameter β that will be defined below, details the construction of the two components of the seed set, S_1 and S_2.

Since Step 5 only terminates when there are no more $(S_1 \cup S_2, r)$-hungry vertices, the algorithm returns a valid contagious set. We now bound the sizes of sets S_1 and S_2.

▶ **Lemma 9.** $|S_1| \leq n\tau_{\max}/\beta$

Proof of Lemma 9. The initial total 1-round hunger is bounded by $n\tau_{\max}$. Each element added to S_1 reduces this quantity by at least β, and since the total 1-round hunger is non-negative, $|S_1|\beta \leq n\tau_{\max}$. ◀

To analyze the size of S_2, we focus on the sub-problem induced by including S_1 in the seed set. The residual problem, Residual(S_1), is an instance of r-round Min-TSS on $V \setminus S_1$ in which, because the process is Markovian, $\tau(v)$ becomes $h_{S_1,1}(v)$. This residual instance has the unusual feature that zero-threshold vertices become active in one round. The degree-reduction step ensures that no vertex in Residual(S_1) has more than β neighbors in $V \setminus S_1$, which leads to the following lemma.

▶ **Lemma 10.** $|S_2| \leq O(|OPT|\beta^{r-1} \log n)$

Proof of Lemma 10. During each iteration of the greedy step, let O be a minimum-cardinality set for which $S_2 \cup O$ is (r-round) contagious for Residual(S_1). We let $\vec{O} = o_1, o_2, o_3, \ldots, o_k$ be an arbitrary, but fixed, ordering over O. Were we to add the elements of $\vec{O}$, in order, to some initially empty set S_3, we would reduce the total hunger $h_r(S_2 \cup S_3)$ from $h_r(S_2)$ down to 0. Therefore, for some o^* in the sequence $\vec{O}$, adding o^* to S_3 causes $h_r(S_2 \cup S_3)$ to decrease by at least $\Delta^* \equiv h_r(S_2)/k$.

We now argue that adding o^* directly to S_2 must also significantly reduce the total amount of hunger amongst (S_2, r)-hungry vertices. Denote the magnitude of this change in total hunger by δ^*, that is, $\delta^* = h_r(S_2) - h_r(S_2 \cup \{o^*\})$. In a t-round activation process, the amount of hunger that can be removed by adding some vertex u to the seed set is bounded above by the number of up-to-length-t simple paths coming out of u. The residual graph has degree bounded by β, and thus the number of such paths is bounded by $1 + \beta + (\beta - 1)^2 + \cdots + (\beta - 1)^t \leq 2\beta^t$.

The addition of o^* to S_3 activates at most δ^* neighbors, each of which may be seen as initiating an $(r-1)$-round activation process. Therefore, by adding o^* to S_3 in the original ordering $\vec{O}$, $h_r(S_2 \cup S_3)$ drops by at most $\delta^* \cdot 2\beta^{r-1}$. By definition, $\Delta^* \leq 2\delta^*\beta^{r-1}$ and hence $\delta^* \geq \Delta^*/(2\beta^{r-1}) = h_r(S_2)/(2k\beta^{r-1})$.

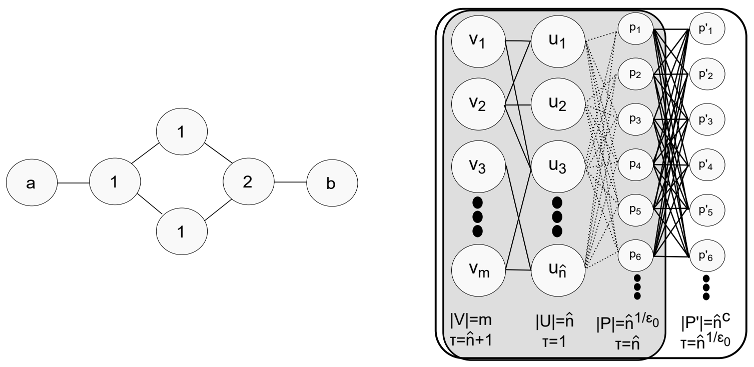


Figure 2 *Left:* This gadget, which has only undirected edges, simulates directed edge $a \to b$. The threshold of each gadget vertex is displayed. An active a "sends" one unit of activation to b, but an active b cannot "send" activation to a. *Right:* To create a hard undirected instance, the unshaded region augments the shaded region, the hard instance of directed MAX-TSS. In the shaded part, all drawn connections are oriented rightwards; in the unshaded part, solid lines represent undirected edges.

We cannot identify o^*: it depends both on the unknown optimal solution and some arbitrary ordering. However, we know that there exists *some* o^* for which

$$h_r(S_2 \cup \{o^*\}) = h_r(S_2) - \delta^* \leq \left(1 - 1/(2\beta^{r-1}k)\right) h_r(S_2)\,. \tag{1}$$

By iterating through each vertex, we can choose (in polynomial time) the vertex o^+ that minimizes $h_r(S_2 \cup \{o^+\})$: this latter expression is clearly also bounded by the right-hand side of inequality (1). Therefore, after $\Theta(k\beta^{r-1}\log n)$ iterations of greedily choosing such o^+, and adding it to S_2, the total hunger $h_r(S_2)$, in the residual graph, drops below 1 and so Algorithm 1 terminates. ◀

Proof of Theorem 8. Given Lemmas 9 and 10, all that remains is to find the optimal choice of β for Algorithm 1. If we knew $|\text{OPT}|$, we would let β be $\Theta([n\tau_{\max}/(|\text{OPT}|\log n)]^{1/r})$, and we would have $|S_1| + |S_2| = O((n\tau_{\max})^{1-1/r}|\text{OPT}|^{1/r}\log^{1/r} n)$, so the ratio of $|S_1| + |S_2|$ to $|\text{OPT}|$ would be

$$O((n\tau_{\max})^{1-1/r}\log^{1/r} n/|\text{OPT}|^{1-1/r}) = O((\tau_{\max}/\tau_{\min})^{1-1/r} n^{1-1/r}\log^{1/r} n)\,, \tag{2}$$

where this "inequality" follows from the fact that $|\text{OPT}| \geq \tau_{\min}$. Although we do not know $|\text{OPT}|$, we can run Algorithm 1 with each β in $1, 2, \dots, n$, and return the best solution; our approximation ratio will be at most the right-hand side of "inequality" (2). ◀

6 Hardness of Undirected Max-TSS

In this section, we show that the hardness of the undirected variant of MAX-TSS matches the $O(n^{1-\varepsilon})$-hardness of the directed variant studied by Kempe, Kleinberg, and Tardos [18].

▶ **Theorem 11.** *For every $\varepsilon > 0$, it is* **NP***-hard to approximate the undirected version of* MAX-TSS *to within an $O(n^{1-\varepsilon})$ multiplicative factor.*

6.1 Revisiting the directed-edge construction

First, we recall the $O(n^{1-\varepsilon_0})$ hardness construction for the directed variant [18], as depicted in the shaded region of Figure 2. Given an instance of the **NP**-hard MAX COVERAGE problem with set system $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, universe of $\hat{n}$ elements $\mathcal{X} = \bigcup_{i=1}^{m} S_i = \{x_1, x_2, \ldots, x_{\hat{n}}\}$ with $|\mathcal{X}| \geq |\mathcal{S}|$, budget k, and coverage goal g, construct a graph containing a vertex $v_i \in V$ for each set S_i, a vertex $u_j \in U$ for each element x_j, and a collection P of $\hat{n}^{1/\varepsilon_0}$ additional "padding" vertices $\{p_i\}$. Whenever $x_j \in S_i$, add a directed edge from v_i to u_j. Next, add a directed edge from each vertex in U to each vertex in P. Every vertex in V has threshold $m + 1$, every vertex in U has threshold 1, and every vertex in P has threshold g.

Now, a seed set that has a vertex y in either U or P is weakly dominated by one that instead has a vertex $v \in V$ with a path to y. If no such v is available, choose an arbitrary unactivated element of V for the seed set. Therefore, consider only solutions in which the seed set S is a subset of V. Since the edges are directed, the vertices in U that are in the activation family are exactly those that have an in-edge from some vertex in the seed set $S \subset V$. Hence the vertex u_i becomes active if and only if the corresponding x_i is in some set S_j for which v_j is active. Thus, it is possible to activate $\geq g$ vertices in U if and only if the Max Coverage instance is satisfiable.

Each $p \in P$ has threshold g and there is the full family of directed edges in $U \times P$. So, if at least g vertices of U become active at some stage, then all $\hat{n}^{1/\varepsilon_0}$ vertices in P become active. If not, then since $m \leq \hat{n}$, no more than $O(\hat{n})$ vertices in the construction become active. Therefore, an $O(\hat{n}^{1/\varepsilon_0}/\hat{n}) = O(\hat{n}^{1/\varepsilon_0 - 1})$-approximation algorithm to Max-TSS can distinguish these two cases, and thus solve the initial Max Coverage instance. As the size of the instance is $n = O(\hat{n}^{1/\varepsilon_0})$, this translates to an $O(n^{1-\varepsilon_0})$ hardness result.

6.2 Translating to undirected edges

Unfortunately, naively replacing the construction's directed edges with undirected edges fails. A single active vertex in P would then activate all of U in a single time step. Instead, replacing each directed edge with the "directed-edge gadget", shown at the left of Figure 2, simulates the previous activation process using only undirected edges. However, these gadgets introduce $O(\hat{n} \cdot \hat{n}^{1/\varepsilon_0})$ extra vertices. To account for the larger problem size, we add a second padding set P' of size $\hat{n}^c$, for some $c \gg 1/\varepsilon_0$ to be chosen later. Each vertex in P', the unshaded region in the right of Figure 2, has threshold $\hat{n}^{1/\varepsilon_0}$.

Finally, we add an undirected edge for each $(p, p') \in P \times P'$. Due to their high thresholds, no vertex in P' will become active before all P are active, so these undirected edges in $P \times P'$ are "safe". As before, including a vertex outside V in the seed set S is dominated by activating one in V, so the analysis in Section 6.1 translates to the construction with set P'. Hence it is **NP**-hard to distinguish $|\mathcal{A}_\infty(S)| = \Omega(\hat{n}^c)$ from $|\mathcal{A}_\infty(S)| = O(\hat{n}^{1+1/\varepsilon_0})$, with the potential activation of gadget vertices being the dominating term in the latter number. As $n = \Theta(\hat{n}^c)$, this translates to an inapproximability factor of $O(n^{1-(1+1/\varepsilon_0)/c})$. For every $\varepsilon > 0$, choosing $c \geq (1 + 1/\varepsilon_0)/\varepsilon$ gives a hardness result of $O(n^{1-\varepsilon})$.

7 Application to Minimum Monotone Satisfying Assignment

Intuitively, the MMSA class of problems asks for the smallest *minterm* of a monotone Boolean function f, i.e., the Boolean vector $\vec{x}$ with $f(\vec{x}) = 1$ that minimizes $|\vec{x}|_1$. However, there is no single agreed definition of MMSA: both the exact description of function f and the measure of quality of a candidate solution $\vec{x}$ have been defined in several ways. The input f has varyingly been given as a monotone formula [17, 13, 8], a monotone circuit [1], and as a circuit that evaluates a monotone function (though it may use $\neg$ gates internally) [21]. The quality of an approximate solution (the "n" in the approximation factor) has also been measured either in terms of the formula/circuit size or in terms of the length of the input vector, in which case the circuit size is assumed to be poly(n). To standardize terminology, we rename MMSA by applying the prefix "MF", "MC", or "NMC" when f is a monotone formula, monotone circuit, or nonmonotone circuit, respectively, and add the superscript "f" or "x" to denote whether it is the description of f or the length of x, respectively, that determines n (in other contexts, previously, subscripts have denoted bounded-depth formulas and circuits).

While each of these models has been shown to give a LabelCover-like hardness of $2^{\log^{1-\varepsilon} n}$, only Umans [21] manages to establish a stronger hardness for the nonmonotone form, namely $n^{1-\varepsilon}$-hardness for both NMC-MMSA^f and NMC-MMSA^x. For MC-MMSA^x, the strongest known hardness results (to our knowledge) conditioned on an established conjecture are only implicitly described by Chlamtac, Dinitz, and Krauthgamer [9]; these lead to a hardness of $n^{3-2\sqrt{2}} \approx n^{.172}$ for the Smallest m-Edge Subgraph problem, assuming some slight modification of Conjecture 1. Despite the weak lower bounds in the case where f is given as a monotone formula/circuit, there is no MMSA problem for which we are aware of a sublinear factor approximation algorithm. A related observation has been previously made by Coja-Oghlan et al. in the context of Target Set Selection [12].

Here, we discuss the scenario in which f is provided as a monotone circuit, and n is either the length of the input or the size of the circuit description. We prove the following conditional results:

▶ **Theorem 12.** *Assuming Conjecture 1, for every $\varepsilon > 0$, it is hard to approximate* MC-MMSA^x *to within an $O(n^{1/2-\varepsilon})$ factor. .*

▶ **Theorem 13.** *Assuming Conjecture 1, for every $\varepsilon > 0$, it is hard to approximate* MC-MMSA^f *to within an $O(n^{1/3-\varepsilon})$.*

We begin with the proof of Theorem 12; the proof of Theorem 13 follows naturally.

Proof of Theorem 12. By the construction in Theorem 3, it is hard (assuming PDS) to approximate constant-round, constant-threshold, degree-$O(\sqrt{n})$ versions of TSS to within a factor of $n^{1/2-\varepsilon}$. Therefore, it suffices to show that we can transform these instances into monotone circuits with size polynomial in n. The reduction is quite simple: for each round $t = 1, 2, \ldots, r$ we have n gates, one for each vertex. The gate corresponding to vertex v is the output of a monotone circuit evaluating the threshold function $\mathrm{Th}_\tau^d(N_{t-1}(v))$, where τ is the threshold of v, d is its degree, and $N_{t-1}(v)$ comprises the gates corresponding to the previous timestep's gates that are in-neighbors of v (for the first timestep, these are just the input gates). Finally, the gates corresponding to the last round all feed into an $\wedge$ gate, forming our output.

The correctness of this circuit is easy to verify. It simulates running the activation process for all r rounds, and checking whether all vertices are active by the end of round r. As the in-degree of each vertex in the RTSS$_r$ instance is $\tilde{O}(\sqrt{n})$, the monotone circuit construction of Friedman [15] requires at most $O(\tau^{12.6}\sqrt{n}\log\sqrt{n}) = O(\sqrt{n}\log n)$ gates. Since we have rn of these circuits, the total number of gates is $O(rn\sqrt{n}\log n) = \mathrm{poly}(n)$. ◀

Proof of Theorem 13. The construction used in the proof of Theorem 12 provides a circuit with $O(m^{3/2}\log m)$ gates for which it is hard to approximate the optimal value to within $m^{1/2-\varepsilon'}$ for every $\varepsilon' > 0$. Choosing n such that $n = m^{3/2}\log m$, for every $\varepsilon > 0$ we have a circuit with n gates whose hardness is $\tilde{O}(n^{(1/2-\varepsilon')/(3/2)}) = O(n^{1/3-\varepsilon})$. ◀

8 Lower Bounds for k-Contagious Set

When a Min-TSS instance contains a contagious set of size k, brute force search (plus simulation of the activation process) can identify such a set in $O(mn^k)$ time, where m is the number of edges in the graph. In this section, we show that improving on this naive approach on directed graphs by even slightly super-linear factors is difficult. Namely, using a construction similar to that in Theorem 6, we show that for $k \geq 3$ an $O(n^{k-\varepsilon})$-time algorithm for k-Contagious Set implies an $O(2^{(1-\varepsilon')n})$-time algorithm for CNF-SAT, violating the Strong Exponential Time Hypothesis (SETH).

▶ **Theorem 14.** *For $k \geq 3$, there is no $O(n^{k-\varepsilon})$-time algorithm for k-*Contagious Set *unless* SETH *is false.*

Proof. We reduce from the k-Dominating Set problem, which has been shown not to admit an $O(n^{k-\varepsilon})$-time algorithm unless SETH is false [19]. Suppose we are given an instance graph $G = (V, E)$ of k-Dominating Set, from which we derive an instance G' of k-Contagious Set. Graph G' contains two vertices, a_v and b_v, for each vertex $v \in V$. A directed edge exists from a_u to b_v whenever either $u = v$ or when v is a neighbor of u in G. Additionally, a directed edge exists from b_u to a_v for every pair u and v, regardless of their adjacency in G. Finally, for all $v \in V$, we set $\tau(a_v) = n$ and $\tau(b_v) = 1$.

It is easy to verify that a size-k dominating set S in G corresponds to a size-k contagious set in G'. The seed set $\{a_v \mid v \in S\}$ activates all of the b vertices, which in turn activates the rest of the a vertices. Conversely, a size-k contagious set S' in G' can be converted into a size-k dominating set for G: $\{v \in V \mid a_v \in S' \text{ or } b_v \in S'\}$. Without loss of generality, we can assume that $S' \subset A$. The choice of a b_v in seed set S' is dominated by the selection of a_v, and no vertex in $A \setminus S'$ could become active unless S' activates all of $\{b\}$ in one round. Hence S' corresponds to a dominating set in G.

Thus, in $O(n^2)$ time, we can reduce finding a k-Dominating Set in G to finding a k-Contagious Set in some G' that has $O(n)$ vertices. Since $k > 2$, it follows that a $O(n^{k-\varepsilon})$-time algorithm for k-Contagious Set on G' implies the existence of such an algorithm for k-Dominating Set, violating SETH. ◀

Acknowledgements. The authors would like to thank Andrew Chester for many fruitful discussions, especially those relating to Section 5.

References

1 Michael Alekhnovich, Sam Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is **NP**-hard to linearly approximate. *The Journal of Symbolic Logic*, 66(01):171–191, 2001.

2 Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 171–180. ACM, 2010.

3 Sanjeev Arora, Boaz Barak, Markus Brunnermeier, and Rong Ge. Computational complexity and information asymmetry in financial products. In *Proceedings of the Innovations in (Theoretical) Computer Science Conference (ICS)*, pages 49–65, 2010.

4 Pranjal Awasthi, Moses Charikar, Kevin A. Lai, and Andrej Risteski. Label optimal regret bounds for online local learning. In *Proceedings of the 28th Conference on Learning Theory (COLT)*, pages 150–166, 2015.

5 Oren Ben-Zwi, Danny Hermelin, Daniel Lokshtanov, and Ilan Newman. Treewidth governs the complexity of target set selection. *Discrete Optimization*, 8(1):87–96, 2011.

6 Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 201–210. ACM, 2010.

7 Ning Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.

8 Ramkumar Chinchani, Duc Ha, Anusha Iyer, Hung Q Ngo, and Shambhu Upadhyaya. On the hardness of approximating the min-hack problem. *Journal of Combinatorial Optimization*, 9(3):295–311, 2005.

9 Eden Chlamtac, Michael Dinitz, and Robert Krauthgamer. Everywhere-sparse spanners via dense subgraphs. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 758–767. IEEE, 2012.

10 Ferdinando Cicalese, Gennaro Cordasco, Luisa Gargano, Martin Milanič, Joseph Peters, and Ugo Vaccaro. Spread of influence in weighted networks under time and budget constraints. *Theoretical Computer Science*, 586:40–58, 2015.

11 Ferdinando Cicalese, Gennaro Cordasco, Luisa Gargano, Martin Milanič, and Ugo Vaccaro. Latency-bounded target set selection in social networks. *Theoretical Computer Science*, 535:1–15, 2014.

12 Amin Coja-Oghlan, Uriel Feige, Michael Krivelevich, and Daniel Reichman. Contagious sets in expanders. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1953–1987. SIAM, 2015.

13 Irit Dinur and Shmuel Safra. On the hardness of approximating label-cover. *Information Processing Letters*, 89(5):247–254, 2004.

14 Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 57–66. ACM, 2001.

15 Joel Friedman. Constructing $O(n \log n)$ size monotone formulae for the kth threshold function of n Boolean variables. *SIAM Journal on Computing*, 15(3):641–654, 1986.

16 Oded Goldreich and Shafi Goldwasser. On the possibility of basing cryptography on the assumption that $\mathbf{P} \neq \mathbf{NP}$., 1998.

17 Michael Goldwasser and Rajeev Motwani. *Intractability of assembly sequencing: Unit disks in the plane.* Springer, 1997.

18 David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146. ACM, 2003.

19 Mihai Pătraşcu and Ryan Williams. On the possibility of faster sat algorithms. In *Proceedings of the 21st Annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 1065–1075. Society for Industrial and Applied Mathematics, 2010.

20 Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 61–70. ACM, 2002.

21 Christopher Umans. Hardness of approximating Σ_2^p minimization problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 465–474. IEEE, 1999.

Approximation Algorithms for Parallel Machine Scheduling with Speed-Up Resources

Lin Chen[1], **Deshi Ye**[2], **and Guochuan Zhang**[3]

1 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
`chenlin198662@gmail.com`

2 **Zhejiang University, College of Computer Science, Hangzhou, China**
`yedeshi@zju.edu.cn`

3 **Zhejiang University, College of Computer Science, Hangzhou, China**
`zgc@zju.edu.cn`

Abstract

We consider the problem of scheduling with renewable speed-up resources. Given m identical machines, n jobs and c different discrete resources, the task is to schedule each job non-preemptively onto one of the machines so as to minimize the makespan. In our problem, a job has its original processing time, which could be reduced by utilizing one of the resources. As resources are different, the amount of the time reduced for each job is different depending on the resource it uses. Once a resource is being used by one job, it can not be used simultaneously by any other job until this job is finished, hence the scheduler should take into account the job-to-machine assignment together with the resource-to-job assignment.

We observe that, the classical unrelated machine scheduling problem is actually a special case of our problem when $m = c$, i.e., the number of resources equals the number of machines. Extending the techniques for the unrelated machine scheduling, we give a 2-approximation algorithm when both m and c are part of the input. We then consider two special cases for the problem, with m or c being a constant, and derive PTASes (Polynomial Time Approximation Schemes) respectively. We also establish the relationship between the two parameters m and c, through which we are able to transform the PTAS for the case when m is constant to the case when c is a constant. The relationship between the two parameters reveals the structure within the problem, and may be of independent interest.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics

Keywords and phrases approximation algorithms, scheduling, linear programming

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.5

1 Introduction

We consider a natural generalization of the classical scheduling problem in which there are multiple different resources available. Each job has an original processing time which may be reduced by utilizing one of the resources. Since resources are different, the amount of the time reduced for each job is different depending on the resource it uses. It is a hard constraint that the usage of the resources does not conflict, that is, once a specific resource is being used by some job, it becomes unavailable to all the other jobs until this job is completed. Consequently a good schedule not only needs to choose the right machine and resource for each job but also needs to sequence jobs on each machine in a proper way such that the usage of each resource does not conflict.

The problem arises naturally in production logistics where a task not only relies on the machine but also on the personnel it is assigned to. It also has its own right from a theoretical point of view. As we will provide details later, this problem is a special case of the general multiprocessor task scheduling problem ($P|set|C_{max}$), which does *not* admit any constant ratio approximation algorithm [2], and meanwhile a generalization of the unrelated machine scheduling problem ($R||C_{max}$), for which a 2-approximation algorithm stands for more than two decades [11].

We give a formal description of our model. There are m parallel identical machines, n jobs and c discrete resources. Each job j has a processing time p_j and has to be processed *non-preemptively* on one of the machines. This processing time might be reduced by utilizing a resource. Specifically, when resource k is allocated to job j then its processing time becomes p_{jk}. At most one resource could be allocated to a job and once a resource, say resource k, is being used by job j, then it can no longer be used by any other job during the time interval where job j is processed. Throughout this paper, we do *not* necessarily require $p_{jk} \leq p_j$. We assume all parameters are taking integral values.

As we have described, in our model jobs could be processed with or without a resource. However, we always assume that each job is processed with a resource unless otherwise specified. Such an assumption causes no loss of generality since we could always introduce m dummy resources (that could not alter the processing time of any job), one for each machine, and jobs scheduled on a machine without a resource could then be viewed as processed with the dummy resource corresponding to this machine. This assumption works for the case that c, the number of resources, is part of the input. For the case that c is a constant, we return to the original assumption that the usage of resources is optional.

Related work. One special case of our problem with $c = 1$ and $m = 2$ is considered in [12], in which an FPTAS (Fully Polynomial Time Approximation Scheme) is derived. Another related problem is considered in [10], in which $c = 1$ again, but the machines are dedicated, i.e., for each job the processing machine is known in advance. For the two-machine case, they prove that the problem is NP-hard but admits an FPTAS. For an arbitrary number of machines, they give a 3/2-approximation algorithm. Moreover, a PTAS is designed for a constant number of machines.

Another closely related model is that a job can be given several resources and yet all resources are identical, so the processing time of each job does not depend on which resource but the number of resources it uses. For this problem on unrelated machines, Grigoriev et al. [3] give a $(3.75 + \varepsilon)$-approximation algorithm. On identical machines, Kellerer [9] gives a $(3.5 + \varepsilon)$-approximation algorithm, which is improved very recently by Jansen, Maack and Rau [4] to an asymptotic PTAS.

Our problem is a generalization of the classical unrelated machine scheduling problem, denoted as $R||C_{max}$, in which each job j has a (machine dependent) processing time p_{ij} if it is processed on machine i. Indeed, if the number of machines is equal to the number of resources, i.e. $m = c$, and $p_j = \infty$ (indeed, it suffices to have $p_j > \sum_{i,j'} p_{ij'}$, as in this case a schedule that does not process job j with any resource is never optimal), then our problem is equivalent to the unrelated machine scheduling problem. To see why, notice that given any feasible solution of our problem, we can rearrange jobs so that all jobs using the same resource, say, k, are scheduled on machine k. By doing so the makespan is not increased, and meanwhile the new solution is a feasible solution of the unrelated machine scheduling problem in which machine k is one of the unrelated machines which processes job j with time p_{jk}. The current best-known approximation ratio for the unrelated scheduling problem is 2 if m is part of the input, whereas no approximation algorithm could achieve a ratio strictly better than 2, assuming $P \neq NP$ [11]. If m is a constant, an FPTAS exists [7] and its current best running time is $O(n) + (\log m/\varepsilon)^{O(m \log m)}$ [5].

Meanwhile, our problem is also a special case of the general multiprocessor task scheduling problem, denoted as $P|set|C_{max}$, in which each task (job), say, j, could be processed simultaneously on multiple machines, and its processing time is $p_{j,S}$ where S is the set of machines we choose to process it. To see why our problem is a special case, we view each resource as a special machine which we call a resource machine, and each job could either be processed on a normal machine with processing time p_j, or processed simultaneously on a normal machine i and some resource machine k, with $p_{j,\{i,k\}} = p_{jk}$. Thus our problem could be transformed to a multiprocessor task scheduling problem with $m + c$ machines. There is a PTAS for the general multiprocessor task scheduling problem if the number of machines is a constant, and no constant ratio approximation algorithm exists if the number of machines is part of the input [2][6]. This result implies that for our problem, if both the number of resources and the number of machines are constants, then there is a PTAS.

Our contribution. We study the scheduling problem with speed-up resources. As we have mentioned, it is an intermediate model between the general model $P|set|C_{max}$ and the classical unrelated machine scheduling $R||C_{max}$. We hope our research could bridge the study of these two well-known models and leads to a better understanding of them.

In this paper, we give the first 2-approximation algorithm when the number of machines m and resources c are both part of the input. We then consider two special cases with either m or c being a constant, and provide PTASes, respectively.

For the general case, we observe that the natural LP (Linear Programming) formulation of the problem has too many constraints, whereas its extreme point solution may split too many jobs which causes the classical rounding technique from [11] inapplicable. To handle this, the key idea is to iteratively remove constraints from the LP. We will iteratively modify the fractional solution such that either we get a new solution with fewer split jobs (which is the same as the traditional rounding), or we get a new solution for which we need fewer constraints to characterize it.

Given the lower bound of 1.5 for the unrelated machine scheduling problem $R||C_{max}$, and hence also for our problem, PTASes are only possible for special cases. We first consider the case when m is a constant and present a PTAS. To achieve this, we first determine (through enumeration) the scheduling of a constant number of jobs, and then handle the scheduling of remaining jobs by formulating it as an LP. We prove that, the LP we construct has a special structure which enforces that only a constant number among its huge number (non-constant) of constraints could become tight and correspond to an extreme point solution. Using this fact we are able to make use of the classical rounding technique from [11] to derive a PTAS.

We then consider the case when c is a constant. We establish an interesting relationship between this special case and the case when m is a constant. Indeed, we show that it suffices to consider solutions where all jobs using resources are scheduled only on $O(c)$ machines. Thus, this special case is a combination of scheduling with resources on $O(c)$ machines, together with the classical scheduling without resources on the remaining $m - O(c)$ machines.

2 General case

In this section, we consider the problem when the number of machines and resources, i.e., m and c, are both part of the input and give a 2-approximation algorithm. Recall that we can assume every job is processed with one resource.

We start with a natural LP formulation of this problem. Let $x_{ijk} = 1$ denote that job j is processed on machine i with resource k, and $x_{ijk} = 0$ otherwise. We first ignore the disjoint condition, i.e., the usage of each resource is not in conflict, and establish the following LP_r.

$$\sum_{i=1}^{m}\sum_{k=1}^{c} x_{ijk} = 1 \qquad \forall j \tag{1a}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} p_{jk}x_{ijk} \le T \qquad \forall k \tag{1b}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{c} p_{jk}x_{ijk} \le T \qquad \forall i \tag{1c}$$

$$0 \le x_{ijk} \le 1 \tag{1d}$$

$$x_{ijk} = 0 \quad \text{if} \quad p_{ik} > T. \tag{1e}$$

Constraint (1a) ensures that every job is scheduled. Constraint (1b) ensures that the total processing time of jobs processed with the same resource k does not exceed T. Constraint (1c) ensures that the total processing time of jobs on each machine does not exceed T. Through binary search we can find the minimum integer $T = T^*$ such that LP_r admits a feasible solution, which is obviously a lower bound on the optimal solution. We denote by x^* the fractional solution of LP_r for $T = T^*$. Our rounding technique tries to make x^* into an integral solution so that (1b) and (1c) could be violated but not much, and the disjoint condition becomes respected, i.e., the disjoint condition which is not met by the LP_r will be achieved via rounding.

We remark that in the classical unrelated machine scheduling problem, the LP relaxation has only $n + m$ constraints, hence in its extreme point solution only m jobs would get split. By re-assigning these jobs to m machines, one per machine, a 2-approximation solution is derived. However, our LP_r has $n + m + c$ constraints. Its extreme point solution may cause $m + c$ jobs to be split, which is too many for carrying out the subsequent re-assigning procedure. To handle this, the key idea of our rounding procedure is to reduce the number of constraints via *well structured* fractional solutions.

In the following we define well structured solutions as well as its *rank*, both of which are crucial for our rounding procedure.

Given any fractional solution x when $T = T^*$, we can compute the fraction of job j processed with resource k through $x_{jk} = \sum_{i=1}^{m} x_{ijk} \in [0, 1]$. We call $\hat{x} = (x_{jk})$ a semi-solution to LP_r.

Obviously it holds for every resource k that $\sum_{j=1}^{n}\sum_{i=1}^{m} p_{jk}x_{ijk} = \sum_{j=1}^{n} p_{jk}x_{jk} \le T^*$. We say resource k is saturated with respect to $\hat{x}$ (and also x) if the equality holds. The number of saturated resources is called the degree of $\hat{x}$ (and x), and denoted as $d(\hat{x})$ $(= d(x))$.

We call $\sum_{j=1}^{n} p_{jk}x_{jk}$ the load of resource k. A semi-solution is called feasible, if the load of each resource is no greater than T^*, and the total load of all resources is no greater than mT^*. Obviously any feasible solution of LP_r implies a feasible semi-solution. On the other hand, any feasible semi-solution also implies a feasible solution of LP_r through the following *Direct Schedule.* (For simplicity we suppose resource 1 to resource $d = d(\hat{x})$ are saturated.)

Direct schedule

1. For $1 \le k \le d$, put (fractions of) jobs using resource k onto machine k.
2. For $k > d$, put (fractions of) jobs using unsaturated resources arbitrarily onto machine $d + 1$ to machine m such that the load of each machine is no greater than T^*.

Consequently, each solution has its corresponding semi-solution, and vice versa.

A semi-solution $\hat{x}$ (and also its corresponding solution x) is called *well structured*, if every job uses at most one unsaturated resource. We have the following lemma.

▶ **Lemma 1.** *Given a feasible semi-solution $\hat{x}$, a feasible well structured semi-solution $\hat{x}'$ can be constructed such that $d(\hat{x}') \geq d(\hat{x})$.*

Proof. For each job j, if it uses two or more unsaturated resources, then $x_{j,k_1} > 0$ and $x_{j,k_2} > 0$ for some unsaturated resources k_1 and k_2. For simplicity we assume the total load of jobs using the two resources are L_1 and L_2 respectively.

Suppose without loss of generality $p_{j,k_1} \leq p_{j,k_2}$, we can choose $\delta = min\{x_{j,k_2}, \frac{T^*-L_1}{p_{j,k_1}}\}$ and replace x_{j,k_1} and x_{j,k_2} with $x_{j,k_1} + \delta$ and $x_{j,k_2} - \delta$ respectively. By doing so either resource k_1 becomes saturated or x_{j,k_2} becomes 0. In both cases the number of unsaturated resources used by job j is decreased by one. Notice that by altering $\hat{x}$ in this way, the total processing time of all jobs does not increase and the load of each resource is still no greater than T^*.

We iteratively apply the above procedure until every job uses at most one unsaturated resource, and a feasible well structured semi-solution $\hat{x}'$ with $d(\hat{x}') \geq d(\hat{x})$ is derived. ◀

Now we are able to define the *rank* of a well structured (semi-)solution.

Again we assume that resource 1 to resource $d(= d(\hat{x}))$ are saturated. A bipartite graph $G(\hat{x}) = (V_1(\hat{x}) \cup V_2(\hat{x}), E(\hat{x}))$ corresponding to $\hat{x}$ is constructed in the following way.

We let $V_1(\hat{x}) = \{J_1, J_2, \cdots, J_n\}$ be the set of job nodes. If $d < m$, then $V_2(\hat{x}) = \{R_0, R_1, R_2, \cdots, R_d\}$ with nodes R_1 to R_d corresponding to the saturated resources, and R_0 corresponding to all the unsaturated resources. Otherwise $d = m$, then there is no unsaturated resources and $V_2(\hat{x}) = \{R_1, R_2, \cdots, R_d\}$. Let $x_{j0} = \sum_{k=d+1}^{c}\sum_{i=1}^{m} x_{ijk} = \sum_{k=d+1}^{c} x_{jk} \in [0, 1]$ if R_0 exists. For $0 \leq k \leq d$, there is an edge $(j, k) \in E(\hat{x})$ if and only if $0 < x_{jk} < 1$.

Additionally, if there are any isolated nodes, we simply remove them (from $V_1(\hat{x})$). This completes the construction of $G(\hat{x})$ for $\hat{x}$.

The rank of a well structured semi-solution $\hat{x}$ is defined as $r(\hat{x}) = |E(\hat{x})| + m - d(\hat{x})$.

The rank will serve as a potential function which allows us to iteratively round an initial feasible solution until a certain criterion is satisfied. Indeed, we have the following.

▶ **Lemma 2.** *Given a well structured semi-solution $\hat{x}$ and its corresponding graph $G(\hat{x}) = (V_1(\hat{x}) \cup V_2(\hat{x}), E(\hat{x}))$, let $G^i(\hat{x}) = (V_1^i(\hat{x}) \cup V_2^i(\hat{x}), E^i(\hat{x}))$ be any of its connected component. If $|E^i(\hat{x})| > |V_1^i(\hat{x})| + |V_2^i(\hat{x})|$, then a well structured solution $\hat{x}'$ of LP_r with a lower rank (i.e. $r(\hat{x}') \leq r(\hat{x}) - 1$) can be constructed in polynomial time.*

Given the above lemma, we are able to show the following key theorem, which directly implies a 2-approximation algorithm.

▶ **Theorem 3.** *Let x^* be the fractional solution of LP_r as we define before. Then an integer solution $x^{IP} = (x_{ijk}^{IP})$ for the following Integer Programming can be derived in polynomial time.*

$$\sum_{i=1}^{m}\sum_{k=1}^{c} x_{ijk} = 1 \qquad \forall j$$

$$\sum_{j=1}^{n}\sum_{i=1}^{m} p_{jk}x_{ijk} \leq T^* + p_{max} \qquad \forall k$$

$$\sum_{j=1}^{n}\sum_{k=1}^{c} p_{jk}x_{ijk} \leq T^* + p_{max} \qquad \forall i$$

$$x_{ijk} \in \{0, 1\}$$

Here $p_{max} = \max_{j,k}\{p_{jk}|x^*_{ijk} \neq 0\}$. And moreover, we could schedule jobs in a proper sequence on each machine so that the disjoint condition is also satisfied. Hence the makespan of the generated schedule is at most twice the optimal solution.

3 The special case with a constant number of machines

In this section, we show that the problem admits a PTAS if the number of machines m is a given constant. Again, we assume that every job is processed with one resource.

Let $\bar{p}_j = \min\{p_{j1}, \cdots, p_{jc}\}$ be the shortest possible processing time of job j and we call it the critical processing time. The resource with which the processing time of job j achieves $\bar{p}_j$ is then called the critical resource of j (if there are multiple such resources, we choose arbitrarily one). We sort jobs so that $\bar{p}_1 \geq \bar{p}_2 \geq \cdots \geq \bar{p}_n$. Consider the first q jobs where q is some constant to be fixed later, we call them critical jobs, and others non-critical jobs.

Notice that we have a 2-approximation algorithm for the general case, thus we can compute some value T such that the makespan of the optimum solution (i.e., OPT) falls in $[T/2, T]$. We provide an algorithm such that given any $t \in [T/2, T]$ and a small positive $\epsilon > 0$, it either determines that $OPT > t$, or produces a feasible schedule with makespan bounded by $t + O(\epsilon T)$.

The basic idea of the algorithm is simple. We first determine (through enumeration) the scheduling of all the critical jobs. For each possible scheduling of the critical jobs, we set up an LP (Linear Programming) for the remaining jobs. If such an LP does not admit a feasible solution, then $OPT > t$. Otherwise we compute its extreme point solution and show that in such a solution only a constant number (depending on q and ϵ) of jobs get split. Finally we show how to construct a feasible schedule based on such a solution.

Configuration schedules. Let $\lambda = 1/\epsilon$ be an integer. Let $ST = \{0, T\epsilon/q, 2T\epsilon/q, \cdots, T + 2T\epsilon\}$ be the set of scaled time points (and hence $|ST| = \lambda q + 2q + 1$). Given a schedule, the processing interval of job j is defined to be the interval (u_j, v_j) such that the processing of j starts at time u_j and ends at time v_j. We say two jobs overlap if they use the same resource and the intersection of their processing interval is nonempty.

A container for a critical job, say, j, is a four-tuple $\vec{v}_j = (i, k_j, a_j, b_j)$ where $1 \leq i \leq m$, $1 \leq k_j \leq c$, $a_j, b_j \in ST$. It implies that job j is processed with resource k_j on machine i during the time window (a_j, b_j) (i.e., its processing interval (u_j, v_j) is a subset of (a_j, b_j)), and furthermore, no other jobs are processed during (a_j, b_j) on machine i.

Obviously there are $mc(\lambda q + 2q + 1)^2$ different kinds of containers. A configuration is then a list of containers for all the critical jobs, namely $(\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_q)$. It can be easily seen that there are at most $m^q c^q (\lambda q + 2q + 1)^{2q}$ different configurations.

A feasible schedule is called a configuration schedule if we can compute a container for each critical job. Notice that this is not always the case since $a_j, b_j \in ST$, and it is possible that any interval (a_j, b_j) during which the critical job j is processed contains some other jobs. Nevertheless, with $O(\epsilon)$-loss we can focus on configuration schedules, as is implied by the following lemma.

▶ **Lemma 4.** *Given a feasible schedule of makespan t, there exists a feasible configuration schedule with makespan no more than $t + 2T\epsilon$.*

Linear Programming for non-critical jobs. Lemma 4 ensures the existence of a configuration schedule whose makespan is bounded by $OPT + 2T\epsilon$. Thus for any $t \in [T/2, T]$, if $t \geq OPT$ then there exists a configuration schedule whose makespan is bounded by $t + 2T\epsilon$.

Recall that there are $\eta \leq m^q c^q(\lambda q + 2q + 1)^{2q}$ different configurations. Let them be CF_1, CF_2, $\cdots$, CF_η. For each configuration, say, CF_κ, the scheduling of critical jobs are fixed. In the following we set up a linear programming $LP_m(CF_\kappa)$ for the remaining jobs.

Suppose according to CF_κ there are $\zeta \leq 2q$ different container points (i.e., the time point when a container starts or ends). We sort them in increasing order as $t_1 < t_2 < \cdots < t_\zeta$. We plug in $t_{\zeta+1} = t + 2T\epsilon$ and $t_0 = 0$.

During each interval (t_i, t_{i+1}) $(0 \leq i \leq \zeta)$, if there is a critical job being processed on a machine, then this machine is called occupied. Otherwise we call it a free machine. Let M_i be the set of free machines during (t_i, t_{i+1}). Similarly during each interval (t_i, t_{i+1}), each resource is either used by a critical job or is not used. Let R_i be the set of resources that are not used during (t_i, t_{i+1}).

Recall that we sort jobs such that $\bar{p}_1 \geq \bar{p}_2 \geq \cdots \geq \bar{p}_n$, and the remaining non-critical jobs are job $q+1$ to job n. We set up a linear programming $LP_m(CF_\kappa)$ as follows.

$$\sum_{i=0}^{\zeta} \sum_{k \in R_i} x_{ijk} = 1, \qquad q+1 \leq j \leq n \tag{2a}$$

$$\sum_{j=q+1}^{n} \sum_{k \in R_i} p_{jk} x_{ijk} \leq (t_{i+1} - t_i)|M_i|, \qquad 0 \leq i \leq \zeta \tag{2b}$$

$$\sum_{j=q+1}^{n} p_{jk} x_{ijk} \leq t_{i+1} - t_i, \qquad 0 \leq i \leq \zeta, k \in R_i \tag{2c}$$

$$x_{ijk} \geq 0, \qquad 0 \leq i \leq \zeta, q+1 \leq j \leq n, k \in R_i \tag{2d}$$

Here x_{ijk} denotes the fraction of job j processed during (t_i, t_{i+1}) with resource k. Constraint (2a) ensures that each non-critical job is scheduled. Since during time interval (t_i, t_{i+1}), only $|M_i|$ machines are free, thus the total load (processing time) of non-critical jobs should not exceed $(t_{i+1} - t_i)|M_i|$, which is implied by (2b). Furthermore, during this interval, the total load of non-critical jobs using any resource $k \in R_i$ is no greater than $t_{i+1} - t_i$ (otherwise the disjoint condition is violated), as is implied by (2c).

As long as $t \geq OPT$, among all the configurations there exists some CF_κ such that $LP_m(CF_\kappa)$ admits a feasible solution. If there is no such a configuration, then we conclude that $t < OPT$. Otherwise, we show a feasible schedule with makespan $t + 3t\epsilon$ could be generated.

▶ **Lemma 5.** *Let x be an extreme point solution of $LP_m(CF_\kappa)$ for some κ, then we have $|\{j | 0 < x_{ijk} < 1 \text{ for some } i, k\}| \leq (m+1)(2q+1)$, i.e., at most $(m+1)(2q+1)$ jobs are split.*

Proof. Suppose there are $\psi \geq n - q$ non-zero variables in the extreme point solution, then they correspond to exact ψ tight constraints among constraints (1), (2) and (3).

Notice that constraints (1) and (2) are composed of $n - q$ and $\zeta + 1$ different inequalities respectively, while constraint (3) is made up of $(\zeta + 1)c$ inequalities. We show that, among the ψ equalities (tight constraints), at most $m(\zeta + 1)$ ones could be from (3). To see why, consider each $0 \leq i \leq \zeta$. For any i, there are at most $|M_i| \leq m$ equalities from (3) since otherwise, the constraint $\sum_{j=q+1}^{n} \sum_{k \in R_i} p_{jk} x_{ijk} \leq (t_{i+1} - t_i)|M_i|$ is violated. Thus $\psi \leq n - q + (m+1)(\zeta+1) \leq n - q + (m+1)(2q+1)$.

Now using a similar argument as [11], we denote μ as the number of jobs getting split (i.e., $x_{ijk} \in (0,1)$ for some i and k), then $2\mu + n - q - \mu \leq \psi \leq n - q + (m+1)(2q+1)$, which completes the proof. ◀

Based on the solution satisfying the above lemma, we show how to generate a near optimal feasible schedule.

First, all the critical jobs are fixed according to CF_κ and we do not need to consider them. Let D be the set of (at most $(m+1)(2q+1)$) split jobs. Temporarily we do not consider them. For each of the remaining non-critical jobs, say, j, there exist some i and k such that $x_{ijk}=1$, implying that job j should be scheduled during (t_i, t_{i+1}) with resource k. Let U_i be the set of all non-critical jobs (excluding jobs in D) to be scheduled during (t_i, t_{i+1}).

Now we aim to schedule jobs of U_i onto $|M_i|$ free machines during (t_i, t_{i+1}). A *preemptive* schedule satisfying the disjoint condition could be constructed as follows: We order jobs in U_i such that jobs using the same resource are adjacent. We pick a free machine of M_i and put jobs one by one onto it according to the job sequence until their total processing time exceeds $t_{i+1}-t_i$. Then the last job is split and on the next machine we start with its remaining fraction, followed by next jobs in the sequence.

Notice that Constraints (2b) and (2c) ensure that any job of U_i has a processing time no more than $t_{i+1}-t_i$, and their total processing time is no more than $|M_i|(t_{i+1}-t_i)$, thus the above method returns a preemptive schedule where at most $|M_i| \le m$ jobs are split. Furthermore, the disjoint condition is satisfied. To see why, consider any resource k. All jobs using this resource are adjacent in the job sequence and their total processing time is no more than $t_{i+1}-t_i$, hence they are scheduled either on one machine or on two machines. If they are on one machine then certainly there is no overlap, otherwise on one machine they are started from t_i and on the other machine they are finished until t_{i+1}, and if there is overlap then their total processing time becomes strictly larger than $t_{i+1}-t_i$, which is a contradiction.

Carrying out the above procedure for each (t_i, t_{i+1}), we derive a preemptive schedule in which at most $m(2q+1)$ jobs get split. We take them out and add them to D. Now it can be easily seen that except for jobs in D, all the other jobs are scheduled integrally during $(0, t+2T\epsilon)$ and the disjoint condition is satisfied.

There are at most $(2m+1)(2q+1)$ jobs in D. Consider the sum of their critical processing times. It remains to show that, there exists a constant q (depending on m and $1/\epsilon$), such that this value is bounded by $T\epsilon$. If this claim holds, then we simply put jobs in D on machine 1 during interval $(t+2T\epsilon, t+3T\epsilon)$ and let each job be processed with its critical resource. A feasible schedule with makespan no more than $t+3T\epsilon$ is derived.

The following lemma from [7] ensures the existence of such a q.

▶ **Lemma 6** ([7]). *Suppose $d_1 \ge d_2 \ge \cdots \ge d_n \ge 0$ is a sequence of real numbers and $D=\sum_{j=1}^n d_j$. Let u, v be nonnegative integers, $\alpha > 0$, and assume that n is sufficiently large (i.e., $n > (\lceil \frac{1}{\alpha}\rceil u+1)(v+1)^{\lceil \frac{1}{\alpha}\rceil}$ suffices). Then, there exists an integer $q=q(u,v,\alpha)$ such that*

$$d_q + d_{q+1} + \cdots + d_{q+u+vq-1} \le \alpha D,$$

$$q \le (v+1)^{\lceil \frac{1}{\alpha}\rceil -1} + u[1+(v+1)+\cdots+(v+1)^{\lceil \frac{1}{\alpha}\rceil -2}].$$

In our problem, $\frac{\sum_{j=1}^n \bar{p}_j}{m} \le OPT \le T$, thus we choose $\alpha = \frac{\epsilon}{m}$, $u=2m+1$ and $v=4m+2$, and derive that $q \le (6m+3)(4m+2)^{\lceil \frac{m}{\epsilon}\rceil}$, which is a constant. Thus we have the following.

▶ **Theorem 7.** *There exists a PTAS for the scheduling with speed-up resources problem when m is a constant.*

4 The special case with a constant number of resources

In this section we assume that each job could be processed with or without a resource. We show that the problem when c is a constant admits a PTAS. The following lemma, which characterize the relationship between the two parameters m and c, is the key to the algorithm.

▶ **Lemma 8.** *Given any positive integer $\lambda = 1/\epsilon$, if there is a feasible solution with makespan T and $m > 3c\lambda$, then there exists a feasible solution with makespan $T(1+\epsilon)$ and all the jobs processed with resources are distributed only on $3c\lambda$ machines.*

Proof idea. The proof is constructive. We only give the main idea here and the reader may refer to the full version of this paper for details. We start with the feasible solution of makespan T and modify it iteratively into the solution satisfying the lemma. During the modification, we only move jobs and do not change the resource each job uses. For simplicity, given a solution, a job processed with resource is called a resource job, and otherwise it is called a non-resource job.

We postpone all jobs by $T\epsilon$ and then divide the time horizon $[0, T(1+\epsilon)]$ equally into $\lambda+1 = 1/\epsilon+1$ sub-intervals, each of length $T\epsilon$. Consider each time point $T\epsilon\eta$ for $1 \leq \eta \leq \lambda$. On each machine, if there is any resource job whose processing interval contains one of these time points, this machine becomes a good machine. It is not difficult to see there are at most $2c\lambda$ good machines. We consider the remaining bad machines. We additionally select $c\lambda$ machines out of them and move all resource jobs of bad machines onto them. This procedure is carried out iteratively. For $1 \leq \eta \leq \lambda$, suppose we have modified the solution so that the following is true: Among all bad machines, there exist $c(\eta-1)$ special machines (called as semi-good machines) such that if the processing of a resource job is finished earlier or at the time $T\epsilon\eta$, then it is either on a good machine or on a semi-good machine. Notice that when $\eta = 1$ this condition is trivially true since we postpone all jobs by $T\epsilon$ and none of them could finish before $T\epsilon$. In step η, we try to additionally select c machines out of the remaining machines (not good or semi-good) and try to move onto them all resource jobs scheduled within $(T\epsilon\eta, T\epsilon(\eta+1))$. Assume for simplicity that there is no job crossing time points $T\epsilon\eta$ and $T\epsilon(\eta+1)$ on the c machines we have selected. The crucial observation is that these c additional machines are neither good nor semi-good, hence *no* resource jobs are scheduled on them *before* $T\epsilon\eta$. Given that we have postponed all jobs by $T\epsilon$, on these c additional machines we could shift back by $T\epsilon$ all the non-resource jobs before $T\epsilon(\eta+1)$, whereas enforcing that during $(T\epsilon\eta, T\epsilon(\eta+1))$ only resource jobs are left on these c machines. Now we could simply take out all the resource jobs scheduled within $(T\epsilon\eta, T\epsilon(\eta+1))$, and let all jobs using the same resource be scheduled on one of the c machines. By doing so the disjoint condition is respected and by adding these additional c machines to semi-good machines, we can continue the above procedure for $\eta+1$. ◀

Let $p_{j0} = p_j$, τ be some constant to be fixed later and $\Lambda = 3c\tau\lambda(\lambda+1)$. Again $\bar{p}_j = \min\{p_{j0}, p_{j1}, \cdots, p_{jc}\}$ is called the critical processing time of job j. We sort all jobs in non-increasing order of their critical processing times. Let T be some integer such that $T/2 \leq OPT \leq T$. A job is called big if $\bar{p}_j > T\epsilon/\tau$, and small otherwise. With $O(\epsilon)$-loss we could round (down) the processing times of big jobs such that p_{jk} is a multiple of $T\epsilon/\Lambda$ (if $p_{jk} > T$ we simply round it to ∞). It is easy to verify that there are $\phi \leq (\lambda\Lambda)^{c+1}$ different kinds of big jobs.

According to Lemma 8, with additional $O(\epsilon)$-loss we may assume that all jobs processed with resources are on the first $3c\lambda$ machines. We call them critical machines and others non-critical machines. With additional $O(\epsilon)$-loss we could further assume that every (rounded) big job j on critical machines has starting and ending times multiples of $T\epsilon/\Lambda$. Let Sol be the solution of makespan $OPT + T \cdot O(\epsilon)$ satisfying all above requirements (the reader may refer to the full version of this paper for a formal proof). In the following we give an algorithm such that given $t \in [T/2, T]$, it either returns a feasible solution of makespan $t + O(T\epsilon)$, or concludes there is no feasible solution of makespan no more than t.

Consider non-critical machines in Sol. We first classify jobs into groups according to p_{j0}. Let $G_l = \{j|(l-1)T\epsilon^2 < p_{j0} \leq lT\epsilon^2, 1 \leq j \leq n\}$ for $\lambda + 1 \leq l \leq \lambda^2$ and $G_\lambda = \{j|p_{j0} \leq T\epsilon\}$. Notice that now we do not round the processing times but only classify jobs into groups. Similar as the traditional parallel machine scheduling problem [1], we use a $(\lambda^2 - \lambda + 2)$-tuple $(\nu_\lambda, \nu_{\lambda+1}, \cdots, \nu_{\lambda^2})$ to represent the jobs scheduled on a non-critical machine. Here ν_l $(\lambda + 1 \leq l \leq \lambda^2)$ is the number of jobs from G_l on this machine. Furthermore, ν_λ is computed in the following way: we first compute the total processing time of jobs from G_λ and let it be ξ, then $\nu_\lambda = \lfloor \frac{\xi}{T\epsilon} \rfloor$. It is easy to verify that there are at most $\lambda^{O(\lambda^2)}$ different kinds of tuples. We list all the tuples as $(\nu_\lambda(i), \cdots, \nu_{\lambda^2}(i))$ for $1 \leq i \leq \gamma = \lambda^{O(\lambda^2)}$. We say a non-critical machine is of type i, if the jobs on it correspond to the tuple $(\nu_\lambda(i), \cdots, \nu_{\lambda^2}(i))$.

Now we define an outline of a feasible schedule. It indicates which big jobs are scheduled on critical machines. Indeed, given a schedule, an outline for it is a ϕ-tuple $\omega = (\omega_1, \omega_2, \cdots, \omega_\phi)$, where ω_i the number of the i-th kind of big jobs that are scheduled on critical machines. Recall that there are at most Λ big jobs on critical machines, there are $(\Lambda + 1)^\phi$ different possible outlines and we could guess out the outline for Sol. Let the outline be O_ι. Let J_b be the set of all big jobs and CR be the set of big jobs on critical machines according to O_ι.

Similar as we did in Section 3, we define a container (i, k_j, a_j, b_j) for a big job j on critical machines, where k_j is its resource, and a_j, b_j are the starting and ending times, which is a multiples of $T\epsilon/\Lambda$. We also define configurations in a similar way. Let $q' \leq |CR|$ be some constant to be determined later. We take out q' jobs in CR with the largest critical processing times and let $W \subset CR$ be the set of them. A configuration is a list of q' containers for the q' jobs in W. Simple calculations show that there are $(\lambda\Lambda)^{O(q')}$ different configurations.

Suppose we guess the correct outline O_ι and configuration CF_κ. According to the configuration, we sort all different container points as $t_1 < t_2 < \cdots t_\zeta$ with $\zeta \leq 2q'$. Again we plug in $t_0 = 0$ and $t_{\zeta+1} = t$ and set up a mixed integer linear programming $MILP(O_\iota, CF_\kappa)$.

$$\sum_{i=0}^{\zeta} \sum_{k \in R_i} x_{ijk} = 1, \qquad j \in CR \setminus W \tag{3a}$$

$$x_{j0} = 1, \qquad j \in J_b \setminus CR \tag{3b}$$

$$x_{j0} + \sum_{i=0}^{\zeta} \sum_{k \in R_i} x_{ijk} = 1, \qquad j \notin W \tag{3c}$$

$$\sum_{j \notin W} \sum_{k \in R_i} p_{jk} x_{ijk} \leq (t_{i+1} - t_i)|M_i|, \qquad 0 \leq i \leq \zeta \tag{3d}$$

$$\sum_{j \notin W} p_{jk} x_{ijk} \leq t_{i+1} - t_i, \qquad 0 \leq i \leq \zeta, k \in R_i \setminus \{0\} \tag{3e}$$

$$z_i = 0 \qquad \text{if} \qquad \sum_{l=\lambda}^{\lambda^2} \nu_l(i)(l-1)T\epsilon^2 \geq t \tag{3f}$$

$$\sum_{i=1}^{\gamma} z_i = m - 3c\lambda \tag{3g}$$

$$\sum_{j\in G_l} x_{j0} = \sum_{i=1}^{\gamma} z_i \nu_l(i), \qquad \lambda+1 \le l \le \lambda^2 \tag{3h}$$

$$\sum_{j\in G_0} x_{j0}p_{j0} \le \sum_{i=1}^{\gamma} z_i \nu_l(i) lT\epsilon, \qquad \lambda \le l \le \lambda^2 \tag{3i}$$

$$x_{ijk} \ge 0, x_{j0} \ge 0 \qquad 0 \le i \le \zeta, j \notin W, k \in R_i \tag{3j}$$

$$z_i \ge 0, z_i \in \mathrm{Z}, \qquad 1 \le i \le \gamma \tag{3k}$$

Here we use similar notations as that of Section 3. Note that the positions of jobs in W are already fixed by CF_κ and we do not need to consider them. R_i is the set of resources that are not used by jobs of W during (t_i, t_{i+1}). Specifically, 0 is taken as a special resource such that if job j is processed without any resource, then it is taken as processed with resource 0. Thus resource 0 is always available and $0 \in R_i$ for any $0 \le i \le \zeta$. M_i is the set of critical machines that are not occupied by jobs of W during (t_i, t_{i+1}) and again we call them as free machines.

We explain the variables used. x_{ijk} is the fraction of job j scheduled during (t_i, t_{i+1}) with resource k. Since during this interval only resources of R_i are available, thus it is only defined for $k \in R_i$. Furthermore, x_{ij0} denotes the fraction of job j scheduled without any resource and as we mention before, it is viewed as processed with resource 0. x_{j0} is the fraction of job j scheduled on non-critical machines. z_i is the number of non-critical machines of type i.

We explain the constraints. Notice that a big job (of J_b) is either on critical machines or on non-critical machines, and this is determined beforehand by O_ι. For $j \in CR$, it should be on critical machines and there are two cases. One is that $j \in W$, then the position of this job is further determined through CF_κ and we do not need to consider it. The other case is $j \in CR \setminus W$, then it should be on critical machines, just as (3a) implies. For big jobs that are not on critical machines, they are on non-critical machines, which is implied by (3b). Constraint (3c) implies that each job should be scheduled either on critical machines or on non-critical machines, and this holds for both big and small jobs.

Constraints (3d) and (3e) are the same with the constraints in LP_m we derive in Section 3. (3d) means the total processing time of jobs scheduled during (t_i, t_{i+1}) on critical machines should not exceed the available times provided by free machines. This is straightforward since the other $3c\lambda - |M_i|$ critical machines are occupied by jobs of W and we can not put jobs on it. (3e) means the total processing time of jobs using resource $k \in R_i$ during (t_i, t_{i+1}) should not exceed $t_{i+1} - t_i$. Notice that 0 should be excluded since it is not a real resource, i.e., jobs processed without resource could be processed at the same time if they are on different machines.

Constraints (3f),(3g),(3h),(3i) are standard constraints. (3f) excludes tuples that are infeasible. (3g) holds as each non-critical machine is of a certain type. Both sides of (3h) equal to the number of jobs in G_l that are scheduled on non-critical machines. Notice that here G_λ is not taken account of since such jobs can be split, just as in the classical scheduling problem. The left side of (3i) calculate the total processing time of jobs in G_l on non-critical machines and the right side is obviously its upper bound.

It can be easily seen that the in the above $MILP$ there is only a constant number of integer variables which is bounded by $\gamma = (\lambda+4)^{\lambda^2-\lambda+1}$, i.e., $2^{O(1/\epsilon^2 \log(1/\epsilon))}$, thus it could be solved in $f(1/\epsilon)poly(n, \log P)$ time using Kannan's algorithm [8]. Here $P = \sum_{j=1}^{n} \bar{p}_j$ is a natural upper bounded for T and $f(1/\epsilon)$ only depends on $1/\epsilon$. Given a feasible solution of the $MILP(O_\iota, CF_\kappa)$, we can show that it could be rounded into an integer solution with an additive loss of $T\epsilon \cdot O(\lambda^2 + cq')$. This is again by observing that once we fix the value of integer variables z_i, there are only a limited number of constraints for the fractional variable x_{ijk}, whereas we get at most $O(\lambda^2 + cq')$ split (small) jobs. Choosing proper q' and τ allows us to bound the overall increase by $O(\epsilon)OPT$. The reader is referred to the full version of this paper for details.

▶ **Theorem 9.** *There is a PTAS for the scheduling with speed-up resources problem when c is a constant.*

Acknowledgements. We thank Janer Chen for pointing out the relationship between the problem we consider and $P|set|C_{max}$ and other useful communications.

References

1 N. Alon, Y. Azar, G.J. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 493–500, 1997. doi:10.1109/SFCS.1975.23.

2 J. Chen and A. Miranda. A polynomial time approximation scheme for general multiprocessor job scheduling. *SIAM journal on computing*, 31(1):1–17, 2001. doi:10.1145/361604.361612.

3 A. Grigoriev, M. Sviridenko, and M. Uetz. Machine scheduling with resource dependent processing times. *Mathematical programming*, 110(1):209–228, 2007. doi:10.1145/361604.361612.

4 K. Jansen, M. Maack, and M. Rau. Approximation schemes for machine scheduling with resource (in-)dependent processing times. In *27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1526–1542, 2016. doi:10.1109/SFCS.1975.23.

5 K. Jansen and M. Mastrolilli. Scheduling unrelated parallelmachines: linear programming strikes back. Technical report, University of Kiel, 2010. Technical Report Bericht-Nr. 1004. doi:10.1109/SFCS.1975.23.

6 K. Jansen and L. Porkolab. General multiprocessor task scheduling: Approximate solutions in linear time. In *Workshop on Algorithms and Data Structures (WADS'99)*, pages 110–121, 1999. doi:10.1109/SFCS.1975.23.

7 K. Jansen and L. Porkolab. Improved Approximation Schemes for Scheduling Unrelated Parallel Machines. *Mathematics of Operations Research*, 26(2):324–338, 2001. doi:10.1145/361604.361612.

8 R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440, 1987. doi:10.1145/361604.361612.

9 H. Kellerer. An approximation algorithm for identical parallel machine scheduling with resource dependent processing times. *Operations Research Letters*, 36(2):157–159, 2008. doi:10.1145/361604.361612.

10 H. Kellerer and V.A. Strusevich. Scheduling parallel dedicated machines with the speeding-up resource. *Naval Research Logistics*, 55(5):377–389, 2008. doi:10.1145/361604.361612.

11 J. K. Lenstra, D. B. Shmoys, and Eva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programing*, 46:259–271, 1990. doi:10.1145/361604.361612.

12 H. Xu, L. Chen, D. Ye, and G. Zhang. Scheduling on two identical machines with a speed-up resource. *Information Processing Letters*, 111(7):831–835, 2011. doi:10.1145/361604.361612.

The Densest k-Subhypergraph Problem

Eden Chlamtáč[*1], **Michael Dinitz**[†2], **Christian Konrad**[‡3], **Guy Kortsarz**[§4], and **George Rabanca**[¶5]

1 **Department of Computer Science, Ben Gurion University, Beersheva, Israel**
`chlamtac@cs.bgu.ac.il.`

2 **Dept. of Computer Science, Johns Hopkins University, Baltimore, MD, USA**
`mdinitz@cs.jhu.edu`

3 **ICE-TCS, School of Computer Science, Reykjavik University, Iceland**
`christiank@ru.is`

4 **Computer Science Department, Rutgers University, Camden, NY, USA**
`guyk@crab.rutgers.edu`

5 **Department of Computer Science, The Graduate Center, CUNY, USA**
`grabanca@gradcenter.cuny.edu`

Abstract

The Densest k-Subgraph (DkS) problem, and its corresponding minimization problem Smallest p-Edge Subgraph (SpES), have come to play a central role in approximation algorithms. This is due both to their practical importance, and their usefulness as a tool for solving and establishing approximation bounds for other problems. These two problems are not well understood, and it is widely believed that they do not an admit a subpolynomial approximation ratio (although the best known hardness results do not rule this out).

In this paper we generalize both DkS and SpES from graphs to hypergraphs. We consider the Densest k-Subhypergraph problem (given a hypergraph (V, E), find a subset $W \subseteq V$ of k vertices so as to maximize the number of hyperedges contained in W) and define the Minimum p-Union problem (given a hypergraph, choose p of the hyperedges so as to minimize the number of vertices in their union). We focus in particular on the case where all hyperedges have size 3, as this is the simplest non-graph setting. For this case we provide an $O(n^{4(4-\sqrt{3})/13+\epsilon}) \leq O(n^{0.697831+\epsilon})$-approximation (for arbitrary constant $\epsilon > 0$) for Densest k-Subhypergraph and an $\tilde{O}(n^{2/5})$-approximation for Minimum p-Union. We also give an $O(\sqrt{m})$-approximation for Minimum p-Union in general hypergraphs. Finally, we examine the interesting special case of interval hypergraphs (instances where the vertices are a subset of the natural numbers and the hyperedges are intervals of the line) and prove that both problems admit an exact polynomial time solution on these instances.

1998 ACM Subject Classification G.2.2 Graph Theory, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Hypergraphs, Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.6

[*] Partially supported by ISF grant 1002/14.
[†] Partially supported by NSF grants 1464239 and 1535887.
[‡] Supported by Icelandic Research Fund grants 120032011 and 152679-051.
[§] Partially supported by NSF grants 1218620 and 1540547.
[¶] Supported by ARL grant W911NF-09-2-0053

1 Introduction

Two of the most important outstanding problems in approximation algorithms are the approximability of the *Densest k-Subgraph* problem (DkS) and its minimization version, the *Smallest p-Edge Subgraph* problem (SpES or min-DkS). In DkS we are given as input a graph $G = (V, E)$ and an integer k, and the goal is to find a subset $V' \subseteq V$ with $|V'| = k$ which maximizes the number of edges in the subgraph of G induced by V'. In the minimization version, SpES, we are given a lower bound p on the number of required edges and the goal is to find a set $V' \subseteq V$ of minimum size so that the subgraph induced by V' has at least p edges. These problems have proved to be extremely useful: for example, a variant of DkS was recently used to obtain a new cryptographic system [3]. The same variant of the DkS problem was shown to be central in understanding financial derivatives [4]. The best-known algorithms for many other problems involve using an algorithm for Densest k-Subgraph or SpES as a black box (e.g. [22, 15, 11]).

Despite decades of work, very little is actually known about these problems. The first approximation ratio for DkS was $O(n^{2/5})$ [18] and was devised in 1993. These days, 23 years later, the best known ratio for the Densest k-Subgraph is $O(n^{1/4+\epsilon})$ for arbitrarily small constant $\epsilon > 0$ [7], and the best known approximation for SpES is $O(n^{3-2\sqrt{2}+\epsilon})$ for arbitrarily small constant $\epsilon > 0$ [9]. Given the slow improvement over 23 years, it is widely believed that DkS and SpES do not admit better than a polynomial approximation ratio. Furthermore, the existing approximation guarantees are tight assuming the recently conjectured hardness of finding a planted dense subgraph in a random graph (for certain parameters) [7, 9]. However, there has been very little progress towards an actual proof of hardness of approximation. It is clear that they are both NP-hard, but that is all that is known under the assumption that $P \neq NP$. Under much stronger complexity assumptions it is known that they cannot be approximated better than some constant [16, 12] or any constant [1], but this is still a long way from the conjectured polynomial hardness.

Based on the believed hardness of DkS and SpES, they have been used many times to give evidence for hardness of approximation. For example, consider the *Steiner k-Forest* problem in which the input is an edge weighted graph, a collection of q pairs $\{s_i, t_i\}_{i=1}^{q}$, and a number $k < q$. The goal is to find a minimum cost subgraph that connects at least k of the pairs. It is immediate to see that SpES is a special case of the Steiner k-forest problem[1], and hence it seems highly unlikely that the Steiner k-Forest problem admits a better than polynomial approximation ratios.

Given the interest in and importance of DkS and SpES, it is somewhat surprising that there has been very little exploration of the equivalent problems in *hypergraphs*. A hypergraph is most simply understood as a collection E of subsets over a universe V of vertices, where each $e \in E$ is called a *hyperedge* (so graphs are the special case when each $e \in E$ has cardinality 2). In general hypergraphs, the obvious extensions of DkS and SpES are quite intuitive. In the *Densest k-Subhypergraph* (DkSH) problem we are given a hypergraph (V, E) and a value k, and the goal is to find a set $W \subseteq V$ of size k that contains the largest number of hyperedges from E. In the *Minimum p-Union* (MpU) problem we are given a hypergraph and a number p, and the goal is to choose p of the hyperedges to minimize the size of their union.

Clearly these problems are at least as hard as the associated problems in graphs, but how much harder are they? Can we design nontrivial approximation algorithms? Can we extend

[1] Given an instance $(G = (V, E), p)$ of SpES, create an instance of Steiner k-Forest on a star with V as the leaves, uniform weights, a demand pair for each edge in E, and $k = p$.

the known algorithms for graphs to the hypergraph setting? Currently, essentially only lower bounds are known: Applebaum [2] showed that they are both hard to approximate to within n^ϵ for some fixed $\epsilon > 0$, assuming that a certain class of one-way functions exist. But it was left as an open problem to design any nontrivial upper bound (see footnote 5 of [2]).

1.1 Our Results

In this paper we provide the first nontrivial upper bounds for these problems. Let n denote the number of vertices and m denote the number of hyperedges in the input hypergraph. Our first result is an approximation for Minimum p-Union in general hypergraphs:

▶ **Theorem 1.** *There is an $O(\sqrt{m})$-approximation for the Minimum p-Union problem.*

We then switch our attention to the low rank case, since this is the setting closest to graphs. In particular, we focus on the 3-*uniform* case, where all hyperedges have size at most 3. In this setting it is relatively straightforward to design an $O(n)$-approximation for Densest k-Subhypergraph, although even this is not entirely trivial (the optimal solution could have size up to k^3 rather than k^2 as in graphs, which would make the trivial algorithm of choosing $k/3$ hyperedges only an $O(n^2)$-approximation rather than an $O(n)$-approximation as in graphs). We show that by very carefully combining a set of algorithms and considering the cases where they are all jointly tight we can significantly improve this approximation, obtaining the following theorem:

▶ **Theorem 2.** *For every constant $\epsilon > 0$, there is an $O(n^{4(4-\sqrt{3})/13+\epsilon}) \leq O(n^{0.697831+\epsilon})$-approximation for the Densest k-Subhypergraph problem on 3-uniform hypergraphs.*

Adapting these ideas to the minimization setting gives an improved bound for Minimum p-Union as well.

▶ **Theorem 3.** *There is an $\tilde{O}(n^{2/5})$-approximation for the Minimum p-Union problem on 3-uniform hypergraphs.*

It is worth noting that any f-approximation for DkSH can be used to give an $\tilde{O}(f)$-approximation for MpU (see Theorem 10), so Theorem 3 gives a significant improvement over this blackbox reduction from Theorem 2.

Finally, we define an interesting special case of Densest k-Subhypergraph and Minimum p-Union that can be solved exactly in polynomial time. Suppose we have an *interval hypergraph*: a hypergraph in which the vertices are a finite subset of $\mathbb{N}$ and each hyperedge is an interval of the real line (restricted to the vertices). Then we show that a dynamic programming algorithm can be used to actually solve our problems.

▶ **Theorem 4.** *Densest k-Subhypergraph and Minimum p-Union can be solved in polynomial time on interval hypergraphs.*

1.2 Related Work

As discussed, the motivation for these problems mostly comes from the associated graph problems, which have been extensively studied and yet are still poorly understood. The Densest k-Subgraph problem was introduced by Kortsarz and Peleg [18], who gave an $O(n^{2/5})$ ratio for the problem. Feige, Kortsarz and Peleg [13] improved the ratio to $O(n^{1/3-\epsilon})$ for ϵ that is roughly 1/60. The current best-known approximation for DkS is $O(n^{1/4+\epsilon})$ for arbitrarily small constant $\epsilon > 0$, due to Bhaskara et al. [7]. For many years the minimization version, SpES, was not considered separately, and it was only relatively recently that the first separation was developed: building on the techniques of [7] but optimizing them for the minimization version, Chlamtáč, Dinitz, and Krauthgamer [9] gave an $O(n^{3-2\sqrt{2}+\epsilon})$-approximation for S$p$ES for arbitrarily small constant $\epsilon > 0$.

While defined slightly differently, DkSH and MpU were introduced earlier by Applebaum [2] in the context of cryptography: he showed that if certain one way functions exist (or that certain pseudorandom generators exist) then DkSH is hard to approximate within n^ϵ for some constant $\epsilon > 0$. Based on this result, DkSH and MpU were used to prove hardness for other problems, such as the k-route cut problem [10]. To the best of our knowledge, though, there has been no previous work on algorithms for these problems.

1.3 Organization

We begin in Section 2 with some preliminaries, showing the basic relationships between the problems. In Section 3 we give our $O(\sqrt{m})$-approximation for MpU in general hypergraphs. We then focus on small-rank hypergraphs, giving an $O(n^{4/5})$-approximation for DkSH on 3-uniform hypergraphs in Section 4, which we then improve to roughly $O(n^{0.698})$ in Section 5. We follow this in Section 6 with our improved bound for MpU on 3-uniform hypergraphs. Finally in Section 7 we show how to solve both problems exactly in polynomial time on interval hypergraphs. We conclude in Section 8 with some open questions for future work.

2 Preliminaries and Notation

A *hypergraph* $H = (V, E)$ consists of a set V (the vertices) together with a collection $E \subseteq 2^V$ (the hyperedges), where each hyperedge is a subset of V. We will typically use $n = |V|$ and $m = |E|$ to denote the number of vertices and hyperedges respectively. The *degree* of a vertex in a hypergraph is the number of hyperedges which contain it. Given a subset $V' \subseteq V$, the subhypergraph of H *induced* by V' is $H[V'] = (V', E_H)$ where $E_H = \{e \in E : e \subseteq V'\}$. We say that H is α-*uniform* if $|e| = \alpha$ for all $e \in E$, and that the *rank* of H is $\max_{e \in E} |e|$ (i.e. the smallest α such that all edges have cardinality at most α). A hyperedge e is *covered* by a set of vertices V' if $e \subseteq V'$.

Given a graph $G = (V, E)$ and a vertex $v \in V$, we use $\Gamma_G(v)$ to denote the set of nodes adjacent to v, and for a subset $V' \subseteq V$ we let $\Gamma_G(V') = \cup_{v \in V'} \Gamma(v)$. If G is clear from context, we will sometimes drop the subscript.

The main problems that we will consider are the following.

▶ **Definition 5.** Given a hypergraph $H = (V, E)$ and an integer k, the *Densest k-Subhypergraph* problem (DkSH) is to find a set $V' \subseteq V$, with $|V'| = k$, such that the number of edges in $H[V']$ is maximized.

▶ **Definition 6.** Given a hypergraph $H = (V, E)$ and an integer p, the *Minimum p-Union* problem (MpU) is to find a set $E' \subseteq E$, with $|E'| = p$, such that $|\cup_{e \in E'} e|$ is minimized.

Note that on 2-uniform hypergraphs, these two problems are the classic graph problems DkS and SpES respectively.

A special class of hypergraphs that we will consider are *interval hypergraphs*, defined as follows.

▶ **Definition 7.** $H = (V, E)$ is an *interval hypergraph* if V is a finite subset of $\mathbb{N}$ and for each $e \in E$ there are values $a_e, b_e \in \mathbb{N}$ such that $e = \{i \in V : a_e \le i \le b_e\}$.

2.1 Relationship Between Problems

We begin by proving some relatively straightforward relationships between the two problems. We first make the obvious observation that a solution for one problem implies a solution for the other.

▶ **Observation 8.** *If there exists a polynomial time algorithm that solves the Densest k-Subhypergraph problem for any k on a hypergraph H, then there exists a polynomial time algorithm that solves the Minimum p-Union problem on the hypergraph H. Similarly, if there is an algorithm that solves MpU on H, then there is an algorithm that solves DkSH on H.*

The relationship is not quite so simple when we are reduced to approximating the problems, but it is relatively straightforward to show that a relationship still exists. This is given by the following lemma, which will also prove to be useful later.

▶ **Lemma 9.** *If there exists an algorithm which in a hypergraph H containing a subhypergraph with k vertices and p hyperedges finds a subhypergraph (V', E') with $|V'| \leq fk$ and $|E'| \geq |V'|p/(kf)$, we can get an $O(f \log p)$-approximation for Min p-Union.*

Since any f-approximation algorithm for Densest k-Subhypergraph satisfies the conditions of the lemma, as an immediate corollary we get the following:

▶ **Theorem 10.** *If there is an f-approximation for Densest k-Subhypergraph, then there is an $O(f \log p)$-approximation for Minimum p-Union.*

Proof of Lemma 9. Let $(H = (V, E), p)$ be an instance of Minimum p-Union, and let $\mathcal{A}$ be an algorithm as described in the lemma. We assume without loss of generality that we know the number of nodes k in the optimal solution (since we can just try all possibilities for k), and hence that there exists a set $V^* \subseteq V$ with $|V^*| = k$ such that V^* covers at least p hyperedges. Initialize $E' = \emptyset$, and consider the following algorithm for Minimum p-Union that repeats the following until $|E'| \geq p$.

1. Let $V' = \mathcal{A}(H, k)$, and let E'' be the hyperedges of H covered by V'.
2. Let $E' \leftarrow E' \cup E''$.
3. Remove E'' from H (remove only the edges, not the corresponding vertices).

We claim that this is an $\tilde{O}(f)$-approximation for Minimum p-Union. Indeed, suppose at iteration i we added x_i vertices, and that at the beginning of the iteration, we had already added $p - p_i$ edges to the solution. In particular, that means that at least p_i of the original hyperedges contained in V^* were not yet removed. This then implies that the number of edges added in iteration i was at least $x_i \cdot p_i/(kf)$. Thus the number of edges we still need to add after iteration i is $p_{i+1} \leq p_i - x_i \cdot p_i/(kf) = p_i(1 - x_i/(kf))$. Thus by induction, after t iterations, the number of hyperedges we need to add is bounded by

$$p_{t+1} \leq p \prod_{i=1}^{t} (1 - x_i/(kf)) \leq p \exp\left(-\sum_{i=1}^{t} x_i/(kf)\right).$$

Thus, as soon as the total number of vertices added exceeds $kf \ln p$ for the first time, the number of edges will exceed p. Since the last iteration adds at most kf vertices, we are done. ◀

A standard argument also shows a (more lossy) reduction in the other direction.

▶ **Theorem 11.** *If there is an f-approximation for Minimum p-Union on α-uniform hypergraphs, then there is an $O(f^\alpha)$-approximation for Densest k-Subhypergraph on α-uniform hypergraphs (when $\alpha = O(1)$).*

Algorithm 1: $2\sqrt{m}$-approximation algorithm for the Minimum p-Union problem

```
Data: Bipartite input graph $G = (E, V, F)$ with $m = |E|$, $n = |V|$, parameter $p$
1  $E' \leftarrow \{\}$;
2  repeat
3      $E'' \leftarrow$ Min-Exp($G[E \setminus E', V]$);
4      if $|E'| + |E''| \leq p$ then
5          $E' \leftarrow E' \cup E''$;
6      else
7          Add arbitrary $p - |E'|$ nodes from $E''$ to $E'$;
8  until $|E'| \geq p - \sqrt{m}$;
9  $E'' \leftarrow$ subset of $p - |E'|$ nodes of $E \setminus E'$ of smallest degree;
10 $E' \leftarrow E' \cup E''$;
11 return $E'$;
```

3 Minimum p-Union in General Hypergraphs

Given a hypergraph $H = (V, E)$, in this section we work with the bipartite *incidence graph* $G = (E, V, F)$ of H, where $F = \{(e, v) \in E \times V \ : \ v \in e\}$. Solving M$p$U on H corresponds to finding a subset $E' \subseteq E$ of p vertices in G of minimum vertex expansion, i.e., E' such that $|\Gamma_G(E')|$ is minimized.

Our algorithm requires a subroutine that returns a subset of vertices of minimum expansion (without the cardinality bound on the set). In other words, we need a polynomial-time algorithm Min-Exp(G) which returns a subset of E so that

$$\frac{|\textsc{Min-Exp}(G)|}{|\Gamma_G(\textsc{Min-Exp}(G))|} \geq \frac{|E'|}{|\Gamma_G(E')|},$$

for every subset $E' \subseteq E$.

Minimally expanding subsets of this kind have previously been used (e.g. in [17, 14]) in communication settings where computation time is disregarded, but in our context we need a polynomial-time algorithm. In Appendices A and B we give two different algorithms for doing this. The first, in Appendix A, uses a reduction to network flows. The second, in Appendix B, is based on a straightforward adaptation of a linear programming approach for the graph case due to Charikar [8]. In order to simplify the presentation, we will for the rest of the section assume that we have such an algorithm and will defer them to the appendices.

In the following, for subsets $E' \subseteq E$ and $V' \subseteq V$, we denote the induced subgraph of G by vertex set $E' \cup V'$ by $G[E', V']$.

In the first phase, our algorithm (Algorithm 1) iteratively adds vertices E'' to an initially empty set E' until E' exceeds the size $p - \sqrt{m}$. The set E'' is a minimally expanding subset in the induced subgraph $G[E \setminus E', V]$. If E'' is large so that $|E' \cup E''| > p$, then an arbitrary subset of E'' is added to E' so that E' has the desired size p. Then, in the second phase, we add the $p - |E'|$ vertices of $E \setminus E'$ of smallest degree to E' (ties broken arbitrarily), and the algorithm returns set E'.

▶ **Theorem 12.** *Algorithm 1 is a $(2\sqrt{m})$-approximation algorithm for MpU.*

Proof. Let $OPT \subseteq E$ be an optimal solution and let $r = |\Gamma_G(OPT)|$. Let E'_i denote the set E' in the beginning of the ith iteration of the repeat loop. Suppose that the algorithm runs in l rounds. Then, E'_{l+1} is the set E' after the last iteration of the loop, but before the nodes selected in Line 9 are added.

Consider an arbitrary iteration $i \leq l$ and let $E'' \leftarrow \textsc{Min-Exp}(G[E \setminus E'_i, V])$ as in the algorithm. Note that by the condition of the loop, we have $|E'_i| \leq p - \sqrt{m}$. Furthermore, we have

$$\frac{|E''|}{|\Gamma_G(E'')|} \geq \frac{|OPT \setminus E'_i|}{|\Gamma_G(|OPT \setminus E'_i|)|} \geq \frac{p - |E'_i|}{r},$$

since E'' is a set of minimum expansion. Then,

$$|\Gamma_G(E'')| \leq \frac{|E''|r}{p - |E'_i|} \leq \frac{|E''|r}{p - p + \sqrt{m}} = \frac{|E''|r}{\sqrt{m}}.$$

Thus, we have $|\Gamma_G(E'_{i+1})| \leq |\Gamma_G(E'_i)| + \frac{|E''|r}{\sqrt{m}}$ (note that this inequality also captures the case when only a subset of E'' is added to E' in Line 7). Now, note that the sets E'' of any two different iterations are disjoint and thus the sizes of the sets E'' of the different iterations sum up to at most m. We thus obtain the bound:

$$|\Gamma_G(E'_{l+1})| \leq \frac{mr}{\sqrt{m}} = \sqrt{m}r.$$

In phase two, we select at most $\sqrt{m}$ vertices E'' of minimum degree in $G[E \setminus E', V]$. Clearly, the maximum degree of these vertices is at most r (if it was larger, then $|\Gamma_G(OPT)|$ would be larger as well) and thus $|\Gamma_G(E'')| \leq \sqrt{m}r$. The neighborhood of the returned set of our algorithm is hence at most $2\sqrt{m}r$ which gives an approximation factor of $2\sqrt{m}$. ◀

4 Densest k-Subhypergraph in 3-uniform hypergraphs

In this section, we consider the Densest k-Subhypergraph problem in 3-uniform hypergraphs. We develop an $O(n^{4/5})$-approximation algorithm here, and show in Section 5 how to improve the approximation factor to $O(n^{0.697831+\epsilon})$, for any $\epsilon > 0$, by replacing one of our subroutines with an algorithm of Bhaskara et al. [7].

Throughout this section, let $H = (V, E)$ be the input 3-uniform hypergraph. Let $K \subseteq V$ denote an optimal solution, i.e., a subset of vertices such that $H[K]$ is a densest k-subhypergraph. The average degree of $H[K]$ is denoted by $d = 3|E(H[K])|/k$. We say that a hyperedge is optimal if it is contained in $H[K]$.

4.1 Overview of our Algorithm

Let $K_1 \subseteq V$ be a set of $k/3$ vertices of largest degree (ties broken arbitrarily), Δ the minimum degree of a node in K_1, and $H' = H[V \setminus K_1]$. Note that the maximum degree in H' is Δ.

Suppose first that at least half of the optimal hyperedges contain at least one vertex of K_1. Then the following lemma shows that we can easily achieve a much better approximation than we are aiming for:

▶ **Lemma 13.** *Suppose that at least half of the optimal hyperedges contain a vertex of K_1. Then we can achieve an $O(n^{1/4+\varepsilon})$ approximation for any $\varepsilon > 0$.*

Proof. By our assumption, there is a set P of optimal hyperedges of size at least $dk/6$ such that every edge in P intersects K_1. Consider two cases.

Case 1: For at least half the edges $e \in P$, we have $|e \cap K_1| \geq 2$. Denote the set of these edges by P'. For every vertex $u \in V$, let its K_1-weight be the number of pairs $\{v, x\}$ such

Algorithm 2: Greedy algorithm for Densest k-Subhypergraph in 3-uniform hypergraphs

```
Data: 3-uniform Hypergraph $H = (V, E)$, parameter $k$, vertex set $K_1 \subseteq V$ of size $k/3$
1 For every $v \in V$, let its $K_1$-degree be $|\{e \in E \mid v \in e, e \cap K_1 \neq \emptyset\}|$;
2 $K_2 \leftarrow$ a set of $k/3$ vertices of highest $K_1$-degree ($K_1$ and $K_2$ may intersect);
3 For any $u \in V$, let its $(K_1, K_2)$-degree be the number of edges of the form $(u, v, x) \in E$
    such that $v \in K_2$ and $x \in K_1$;
4 $K_3 \leftarrow$ a set of $k/3$ vertices of highest $(K_1, K_2)$-degree. ($K_3$ may intersect $K_1$ and/or
    $K_2$);
5 return $K_1 \cup K_2 \cup K_3$;
```

$v, x \in K_1$ and $\{u, v, x\}$ is a hyperedge. Then by our assumption, the vertices in K have average K_1-weight at least $|P'|/k \geq d/12$. Choosing $2k/3$ vertices greedily (by maximum K_1-weight) gives (along with K_1) a k-subhypergraph with at least $dk/18$ hyperedges.

Case 2: $P'' = P \setminus P'$ contains at least half the hyperedges in P. Note that $|e \cap K_1| = 1$ for every $e \in P''$. For every pair of vertices $u, v \in V \setminus K_1$, let its K_1-weight be the number of vertices $x \in K_1$ such that $\{u, v, x\}$ is a hyperedge, and let G be the graph on vertices $V \setminus K_1$ with these edge weights. Then any k'-subgraph of G with total edge weight w corresponds to a $(|K_1| + k')$-subhypergraph of H with at least w hyperedges, and in particular, G contains a k-subgraph with average weighted degree at least $2|P''|/k \geq d/6$, which can be easily pruned (randomly or greedily) down to a $2k/3$-subgraph with average weighted degree $\Omega(d)$. Thus we can run the Densest k-Subgraph approximation algorithm of Bhaskara et al. [7][2], and find a $2k/3$-subgraph of G with total weight at least $kd/n^{1/4+\varepsilon}$, which in turn gives a $(|K_1| + 2k/3 =)k$-subhypergraph of H with a corresponding number of hyperedges. ◀

In the more difficult case, at least half of the optimal hyperedges are fully contained in H'. Exploiting the fact that the maximum degree in H' is Δ and trading off multiple algorithms, we show in the following subsection how to obtain an $O(n^{\frac{4}{5}})$-approximation algorithm in this case.

4.2 An $O(n^{4/5})$-approximation

We start with a greedy algorithm similar to the greedy algorithm commonly used for Densest k-Subgraph [18, 13, 7].

Algorithm 2 selects a subset K_2 of $k/3$ vertices v with largest K_1-degree, i.e., the number of hyperedges incident to v that contain at least one vertex of K_1. Then, a subset K_3 of $k/3$ vertices w with largest (K_1, K_2)-degree is selected, where the (K_1, K_2)-degree of w is the number of hyperedges containing w of the form $\{w, x, y\}$ with $x \in K_1$ and $y \in K_2$. Note that the sets K_1, K_2 and K_3 are not necessarily disjoint and the returned set may thus be smaller than k.

The following lemma gives a lower bound on the average degree guaranteed by this algorithm. It is a straightforward extension of similar algorithms for graphs.

► **Lemma 14.** *Algorithm 2 returns a k-subhypergraph with average degree $\Omega(\Delta k^2/n^2)$.*

[2] Strictly speaking, the algorithm in [7] is defined for unweighted graphs, but one can easily adapt it by partitioning the edges into $O(\log n)$ sets with similar edge weights, and running the algorithm separately on every set of edges, thus losing only an additional $O(\log n)$ factor in the approximation.

Algorithm 3: A neighborhood-based algorithm for Densest k-Subhypergraph in 3-uniform hypergraphs

```
Data: 3-uniform Hypergraph $H' = (V', E')$ and parameter $k$.
1 foreach vertex $v \in V$ do
2     $G_v \leftarrow (V \setminus \{v\}, \{(u,x) \mid (v,u,x) \in E\})$;
3     foreach integer $\hat{d} \in [k-1]$ do
4         $G_v^{\hat{d}} \leftarrow G_v$;
5         while there exists a vertex $u$ in $G_v^{\hat{d}}$ of degree $< \hat{d}$ do
6             delete $u$ from $G_v^{\hat{d}}$;
7         $S_v^{\hat{d}} \leftarrow$ a set of $(k-1)/2$ vertices with highest degree in $G_v^{\hat{d}}$;
8         $T_v^{\hat{d}} \leftarrow$ a set of $(k-1)/2$ vertices with the most neighbors in $S_v^{\hat{d}}$;
9 return The densest among all subhypergraphs $H'[\{v\} \cup S_v^{\hat{d}} \cup T_v^{\hat{d}}]$ over all choices of $v, \hat{d}$;
```

Proof. By choice of K_1 and definition of Δ, every vertex in K_1 has degree at least Δ, and so the total number of edges containing vertices in K_1 is at least $\Delta|K_1|/3 = \Delta k/9$ (since we could potentially be double-counting or triple-counting some edges).

If we were to choose n vertices for K_2, there would be at least $\Delta k/9$ edges containing both a vertex in K_1 and a vertex in K_2 (as noted above). Choosing $k/3$ vertices greedily out of n yields a set K_2 such that there are at least $\Delta k/9 \cdot (k/3)/n = \Delta k^2/(27n)$ such edges.

Finally, choosing the $k/3$ vertices with the largest contribution (out of n) for K_3 ensures that there will be at least $\Delta k^2/(27n) \cdot (k/3)/n = \Omega(\Delta k^3/n^2)$ edges in $E \cap K_1 \times K_2 \times K_3$, giving average degree $\Omega(\Delta k^2/n^2)$. ◀

We now offer a second algorithm, which acts on H' and is based on neighborhoods of vertices.

Algorithm 3 exploits the bound on the maximum degree in H' to find a dense hypergraph inside the neighborhood of any vertex of degree $\Omega(d)$ in K, by considering the neighborhood of a vertex as a graph. Pruning low-degree vertices in this graph (which would not contribute many hyperedges to K) helps reduce the size of the graph, and makes it easier to find a slightly denser subgraph. Since the vertices of K and their degrees are not known, the algorithm tries all possible vertices.

▶ **Lemma 15.** *If H' contains a k-subhypergraph with average degree $d' = \Omega(d)$, then Algorithm 3 returns a k-subhypergraph with average degree $\Omega(d^2/(\Delta k))$.*

Proof. Since at the end of the algorithm we take the densest induced subhypergraph of H' (among the various choices), it suffices to show that there is some choice of v and $\hat{d}$ which gives this guarantee. So let v be an arbitrary vertex in K with degree (in K) at least d'. We know that G_v contains a subgraph with at most k vertices and at least d' edges, so its average degree is at least $2d'/k$. Setting $\hat{d} = d'/(2k)$, we know that the pruning procedure can remove at most $k \cdot d'/(2k) = d'/2$ out of the d' edges in this subgraph, so the subgraph still retains at least $d'/2$ edges. On the other hand, we know that G_v has at most Δ edges (since we've assumed the maximum degree in H' is at most Δ), and therefore, the same holds for the graph $G_v^{\hat{d}}$, in which the minimum degree is now at least $d'/2k$. This means that $G_v^{\hat{d}}$ has at most $2\Delta/(d'/2k) = O(\Delta k/d)$ vertices.

Since there exists a k-subgraph of $G_v^{\hat{d}}$ with $\Omega(d)$ edges, the greedy choice of $S_v^{\hat{d}}$ must give some set in which at least $\Omega(d)$ edges are incident. The greedy choice of $T_v^{\hat{d}}$ then reduces the lower bound on the number of edges by a $((k-1)/2)/|V(G_v^{\hat{d}})| = \Omega(d/\Delta)$ factor, giving us $\Omega(d^2/\Delta)$ edges. However, by the definition of G_v, together with v these edges correspond to hyperedges in H'. Thus, the algorithm returns a k-subhypergraph with $\Omega(d^2/\Delta)$ hyperedges, or average degree $d^2/(\Delta k)$. ◀

Combining the various algorithms we've seen with a trivial algorithm and choosing the best one gives us the following guarantee:

▶ **Theorem 16.** *There is an $O(n^{4/5})$-approximation for Dense k-Subhypergraph in 3-uniform hypergraphs.*

Proof. By Lemma 13, if at least half the optimal edges intersect K_1, then we can achieve a significantly better approximation (namely, $n^{1/4+\varepsilon}$). Thus, from now on let us assume this is not the case. That is, H' still contains a k-subhypergraph with average degree $\Omega(d)$. Again, recall that the maximum degree in H' is at most Δ.

By Lemma 14, Algorithm 2 gives us a k-subhypergraph with average degree $d_1 = \Omega(\Delta k^2/n^2)$. On the other hand, applying Algorithm 3 to H' will give us a k-subhypergraph with average degree $d_2 = \Omega(d^2/(\Delta k))$ by Lemma 15.

Finally, we could choose $k/3$ arbitrary edges in H and the subhypergraph induced on the vertices they span, giving us average degree $d_3 \geq 1$. Thus, the best of the three will give us a k-subhypergraph with average degree at least

$$\max\{d_1, d_2, d_3\} \geq (d_1^2 d_2^2 d_3)^{1/5} = \Omega((\Delta^2 k^4/n^4 \cdot d^4/(\Delta^2 k^2))^{1/5}) = d \cdot \Omega((k^2/d)^{1/5}/n^{4/5}).$$

Since we must have $k^2/d \geq 1$, the above gives an $O(n^{4/5})$-approximation. ◀

5 An improved approximation for 3-uniform Densest k-Subhypergraph

In Section 4 we gave an $O(n^{4/5})$ approximation which combined a greedy algorithm with Algorithm 3, which looked for a dense subgraph inside a graph defined by the neighborhood of a vertex in H. To find this dense subgraph, we used a very simple greedy approach. However, we have at our disposal more sophisticated algorithms, such as that of Bhaskara et al. [7]. One way to state the result in that paper (see Bhaskara's PhD thesis for details on this version [6]) is as follows:

▶ **Theorem 17.** *In any n-vertex graph G, for any $\alpha \in [0,1]$, if $k = n^\alpha$, then Densest k-Subgraph in G can be approximated within an $n^\varepsilon k^{1-\alpha}$ factor in time $n^{O(1/\varepsilon)}$ for any $\varepsilon > 0$.*

The $n^{1/4+\varepsilon}$ guarantee of [7] follows since for any $\alpha \in [0,1]$, we have $k^{1-\alpha} = n^{\alpha(1-\alpha)} \leq n^{1/4}$.

Using this guarantee instead of the simple greedy algorithm for DkS, we get the following improved algorithm for 3-uniform Densest k-Subhypergraph:

The approximation guarantee in this final algorithm is given by the following lemma:

▶ **Lemma 18.** *Let H' be an n-vertex 3-uniform hypergraph with maximum degree $\leq \Delta$, containing a k-subhypergraph of average degree d', and let α, β be such that $k = n^\alpha$ and $\Delta k/d' = n^\beta$. Then Algorithm 4 returns a k-subhypergraph of H of average degree*

$$\Omega\left(\frac{d'}{n^{\varepsilon+\alpha(2-\alpha/\min\{\beta,1\})}}\right).$$

Algorithm 4: A DkS-based algorithm for Densest k-Subhypergraph in 3-uniform hypergraphs

```
   Data: 3-uniform Hypergraph $H' = (V', E')$ and parameters $k$ and $\varepsilon > 0$.
1  foreach vertex $v \in V$ do
2      $G_v \leftarrow (V \setminus \{v\}, \{(u,x) \mid (v,u,v) \in E\})$;
3      foreach integer $\hat{d} \in [k-1]$ do
4          $G_v^{\hat{d}} \leftarrow G_v$;
5          while there exists a vertex $u$ in $G_v^{\hat{d}}$ of degree $< \hat{d}$ do
6              Delete $u$ from $G_v^{\hat{d}}$;
7          $K_v^{\hat{d}} \leftarrow$ the vertex set returned by the algorithm of Bhaskara et al. [7] on the
           graph $G_v^{\hat{d}}$ with parameters $k-1$ and $\varepsilon$;
8  return The densest among all subhypergraphs $H'[\{v\} \cup K_v^{\hat{d}}]$ over all choices of $v, \hat{d}$;
```

Proof. As in the proof of Lemma 15, we can deduce that for at least some choice of v and $\hat{d}$, the graph $G_v^{\hat{d}}$ has at most $\min\{n, O(\Delta k/d')\} = O(n^{\min\{1,\beta\}})$ vertices and contains a k-subgraph with average degree $\Omega(d'/k)$.

By Theorem 17, since $k = n^\alpha = \Omega(|V(G_v^{\hat{d}})|^{\alpha/\min\{1,\beta\}})$, the algorithm of [7] will return a $(k-1)$-subgraph of $G_v^{\hat{d}}$ with average degree

$$\Omega\left(\frac{d'/k}{n^\varepsilon k^{1-\alpha/\min\{\beta,1\}}}\right) = \Omega\left(\frac{d'}{n^{\varepsilon+\alpha(2-\alpha/\min\{\beta,1\})}}\right).$$

As noted in the proof of Lemma 15, this corresponds to a k-subhypergraph of H' with the same guarantee. ◀

▶ Remark. In the notation of Lemma 18 we have $\Delta/d' = n^{\beta-\alpha}$ which implies that $\beta \geq \alpha$ (since $\Delta \geq d'$).

Trading off the various algorithms we have seen, we can now prove the guarantee stated in Theorem 2.

▶ **Theorem 19** (Theorem 2 restated)**.** *For every constant $\varepsilon > 0$, there exists a polynomial time algorithm that achieves an $O(n^{4(4-\sqrt{3})/13+\varepsilon}) \leq O(n^{0.697831+\varepsilon})$-approximation for Densest k-Subhypergraph in 3-uniform hypergraphs.*

Proof. By Lemma 13, if at least half the optimal edges intersect K_1, then we can achieve a significantly better approximation (namely, $n^{1/4+\varepsilon}$). Thus, from now on let us assume this is not the case. That is, H' still contains a k-subhypergraph with average degree $\Omega(d)$. Again, recall that the maximum degree in H' is at most Δ.

As before, let α, β be such that $k = n^\alpha$ and $\Delta k/d = n^\beta$. By Lemma 14, Algorithm 2 gives us a k-subhypergraph with average degree

$$d_1 = \Omega(\Delta k^2/n^2) = \Omega\left(\frac{d}{(d/\Delta)n^2/k^2}\right) = \Omega\left(\frac{d}{n^{\alpha-\beta}n^{2-2\alpha}}\right) = \Omega\left(\frac{d}{n^{2-\alpha-\beta}}\right).$$

On the other hand, by Lemma 18, Algorithm 4 to H' will give us a k-subhypergraph with average degree

$$d_2 = \Omega\left(\frac{d}{n^{\varepsilon+\alpha(2-\alpha/\min\{\beta,1\})}}\right).$$

Let us analyze the guarantee given by the best of Algorithm 2 and Algorithm 4. First, consider the case of $\beta > 1$. In this case, taking the best of the two gives us approximation ratio at most $n^{\varepsilon+\min\{2-\alpha-\beta,\alpha(2-\alpha)\}} \leq n^{\varepsilon+\min\{1-\alpha,\alpha(2-\alpha)\}}$. It is easy to check that this minimum is maximized when $\alpha = (3-\sqrt{5})/2$ giving approximation ratio $n^{(\sqrt{5}-1)/2+\varepsilon} \leq n^{0.618034+\varepsilon}$, which is even better than our claim.

Now suppose $\beta \leq 1$. In this case, the approximation guarantee is $n^{\varepsilon+\min\{h_1,h_2\}}$, where $h_1 = 2-\alpha-\beta$ and $h_2 = \alpha(2-\alpha/\beta)$. If $\alpha \geq 2/3$, then it can be checked that we always have $h_1 \leq h_2$ for any $\beta \in [\alpha, 1]$, in which case we have approximation factor at most $n^{\varepsilon+2-2/3-2/3} = n^{2/3+\varepsilon}$, which is again better than our claim. On the other hand, if $\alpha \leq (3-\sqrt{5})/2$, then $h_2 \leq h_1$ for any $\beta \leq 1$, and so for this range of α we get approximation factor at most $n^{\varepsilon+\alpha(2-\alpha)} \leq n^{(\sqrt{5}-1)/2}$, which as we've noted is also better than our claim. Finally, if $\alpha \in ((3-\sqrt{5})/2, 2/3)$ then a straightforward calculation shows that

$$\min\{h_1,h_2\} = \begin{cases} h_1 & \text{if } \beta \geq 1 - \frac{3\alpha}{2} + \sqrt{1-3\alpha+13\alpha^2/4} \\ h_2 & \text{otherwise,} \end{cases}$$

and that the value of $\min\{h_1,h_2\}$ is maximized at this threshold value of β. And so for α in this range we have $\min\{h_1,h_2\} \leq 1 + \alpha/2 - \sqrt{1-3\alpha+13\alpha^2/4}$, which is maximized at $\alpha = \frac{18+2\sqrt{3}}{39} \approx 0.55$, giving approximation ratio $n^{\varepsilon+4(4-\sqrt{3})/13}$. ◀

6 Minimum p-Union in 3-uniform hypergraphs

In this section we explore Minimum p-Union (the minimization version of Densest k-Subhypergraph), and give the following guarantee:

▶ **Theorem 20.** *There is an $\tilde{O}(n^{2/5})$-approximation algorithm for Minimum p-Union in 3-uniform hypergraphs.*

Note that this is significantly better than the $n^{0.69\cdots}$-approximation we would get by reducing the problem to Densest k-Subhypergraph via Theorem 10 and applying the approximation algorithm from Theorem 2.

In this problem, we are given a 3-uniform hypergraph $H = (V, E)$, and a parameter p, the number of hyperedges that we want to find. Let us assume that the optimal solution, $P \subseteq E$, has k vertices (i.e. $|\cup_{e\in P} e| = k$). We do not know k, but the algorithm can try every possible value of $k = 1, \ldots, n$, and output the best solution. Thus, we assume that k is known, in which case the average degree in the optimum solution is $d = 3p/k$.

Recall that it is not necessary to get p edges in one shot. By Lemma 9, it is enough to find any subhypergraph of size at most $kn^{2/5}$ with average degree at least $\Omega(d/n^{2/5})$.

We follow along the lines of DkSH by choosing vertex set K_1 to be the $kn^{2/5}$ vertices of largest degree. The following lemma (corresponding to Lemma 13 for DkSH) shows that if at least half the edges in P intersect K_1, then by Lemma 9 we are done.

▶ **Lemma 21.** *Suppose that at least half of the optimal edges contain a vertex of K_1. Then we can find a subhypergraph with at most $O(kn^{2/5})$ vertices and average degree at least $\Omega(d/n^{2/5})$.*

Proof. By our assumption, there is a set of optimal hyperedges $P' \subset P$ of size at least $dk/6$ such that every edge in P' intersects K_1.

As in the proof of Lemma 13, if at least half the edges in P' intersect K_1 in more than one vertex, then we can easily recover a set of k vertices which along with K_1 contain at least $\Omega(p) = \Omega(kd)$ hyperedges. Since $|K_1| = kn^{2/5}$, this subgraph has $O(kn^{2/5})$ vertices and average degree $\Omega(d/n^{2/5})$ as required.

Thus, we may assume that at least half the edges in P' intersect K_1 in exactly one vertex. Then again as in Lemma 13, we define a graph G on vertices $V \setminus K_1$ where every pair of vertices $u, v \in V \setminus K_1$ is an edge with weight $|\{x \in K_1 \mid (u, v, x) \in E\}|$. Once again, subgraphs of G with total edge weight w correspond to a subhypergraphs of H with at least w edges, and in particular, G contains a k-subgraph with average weighted degree at least $\Omega(d)$. Thus running the SpES approximation of [9] (or more precisely, the weighted version [11]), gives a subgraph with at most kf vertices and total edge weight at least $\Omega(kd)$ for some $f = n^{0.17+\varepsilon}$ (which is well below $n^{2/5}$). Once again, the corresponding subhypergraph has at most $|K_1| + kf = O(kn^{2/5})$ vertices, and so the average degree is at least $\Omega(d/n^{2/5})$ as required. ◀

Thus, we will assume from now on that at least half of the hyperedges in P *do not* contain at least one vertex from K_1, i.e. that $H' = H[V \setminus K_1]$ still contains at least half the hyperedges in P.

As with DkSH, we now proceed with a greedy algorithm. Starting with the same vertex set K_1 defined above, it follows from Lemma 14 that if we run Algorithm 2 on H with parameter $n^{2/5}k$, then we get a subhypergraph on $O(kn^{2/5})$ vertices induced on sets K_1, K_2, K_3 such that if the minimum degree in K_1 (which bounds the maximum degree in $V \setminus K_1$) is Δ, then the subhypergraph has average degree $\Omega(\Delta k^2 n^{4/5}/n^2)$. The total number of hyperedges in this subhypergraph is $\Omega(\Delta k^3 n^{6/5}/n^2) = \Omega(\Delta k^3/n^{4/5})$. If this is at least $p = dk/3$, then we are done. Thus, we will assume from now on that $\Delta k^3/n^{4/5} = O(dk)$, that is

$$\Delta = O\left(\frac{dn^{4/5}}{k^2}\right). \tag{1}$$

We reuse Algorithm 3 on H', which gives us the following guarantee:

▶ **Lemma 22.** *Applying Algorithm 3 to the above hypergraph H' with parameter*

$$\hat{k} = \frac{k\sqrt{p\Delta}}{d} = \sqrt{\frac{k^3\Delta}{3d}}$$

returns a subhypergraph with at most kf vertices and average degree at least d/f for some

$$f = O(\max\{k, n^{2/5}/\sqrt{k}\}).$$

Proof. As in the proof of Lemma 15, we can deduce that for at least some choice of v and $\hat{d}$, the graph $G_v^{\hat{d}}$ has at most $O(\Delta k/d)$ vertices and has minimum degree at least $\Omega(d/k)$.

Note that we may not even have $\hat{k}$ vertices in $G_v^{\hat{d}}$. If we do have at least $\hat{k}$ vertices, then the greedy choice of $S_v^{\hat{d}}$ gives us $\Omega(\hat{k}d/k)$ edges incident in the set (in fact, any choice of $\Omega(\hat{k})$ vertices would do). The greedy choice of $T_v^{\hat{d}}$ then reduces the number of edges by (in the worst case) a $\hat{k}/(\Delta k/d)$-factor, giving us a total number of edges

$$\Omega\left(\frac{\hat{k}^2 d^2}{\Delta k^2}\right) = \Omega(p).$$

Thus, in this case, we only need to bound the size of the subgraph. By (1), we can bound $\hat{k}$ as follows:

$$\hat{k} = \sqrt{\frac{k^3\Delta}{3d}} = O\left(\sqrt{\frac{dn^{4/5}}{k^2} \cdot \frac{k^3}{d}}\right) = O\left(k \cdot \frac{n^{2/5}}{\sqrt{k}}\right),$$

which proves the lemma for this case.

If we do not have $\hat{k}$ vertices in $G_v^{\hat{d}}$, then the algorithm simply returns $G_v^{\hat{d}}$ itself, which has at most $\hat{k} = O(k \cdot n^{2/5}/\sqrt{k})$ vertices and average degree at least $\Omega(d/k)$, as required.

As noted in the proof of Lemma 15, this corresponds to a subhypergraph of H' with the same guarantee. ◀

We can now prove the main theorem.

Proof of Theorem 20. By Lemma 22 and Lemma 9, to prove the theorem it suffices to show that $\max\{k, n^{2/5}/\sqrt{k}\} = O(n^{2/5})$. Since clearly $n^{2/5}/\sqrt{k} \leq n^{2/5}$, let us consider the parameter k. By definition of d and Δ, we clearly have $d \leq \Delta$, thus, by (1) we have

$$d \leq \Delta = O\left(\frac{dn^{4/5}}{k^2}\right)$$

which implies $k = O(n^{2/5})$, and so the theorem follows. ◀

7 Interval Hypergraphs

We show now that DkS and MpU can be solved in polynomial time on interval hypergraphs. We only give an algorithm for MpU; a similar algorithm for DkS follows then from Observation 8.

As defined in Section 2, a hypergraph $H = (V, E)$ is an interval hypergraph, if $V \subseteq \mathbb{N}$ and for each $e \in E$ there are integers a_e, b_e such that $e = \{i \in V : a_e \leq i \leq b_e\}$. Solving M$p$U on H can be interpreted as finding p intervals with minimum joint support.

▶ **Theorem 23.** Minimum p-Union *is solvable in polynomial time on interval hypergraphs.*

Proof. Let $b_1, \ldots, b_m$ be the largest elements in hyperedges $e_1, \ldots, e_m$ respectively, and assume that $b_i \leq b_j$ for any $i < j$. Similarly let $a_1, \ldots, a_m$ be the smallest elements in $e_1, \ldots, e_m$ respectively.

We present a dynamic programming algorithm which calculates for each $j \leq i$ the optimal solution to an instance of Minimum p-Union on the hyperedges $e_1, \ldots, e_i$ with $p = j$ under the constraint that e_i belongs to the solution. Let $A[i, j]$ store the value of this optimal solution. Assume that the values of A have been computed for all i', j' with $j' \leq i' < i$. We show how to compute $A[i, j]$ for any $j \leq i$.

We partition the hyperedges $e_1, \ldots, e_i$ in three sets A_i, B_i, C_i with A_i containing all hyperedges disjoint from e_i, B_i containing all hyperedges intersecting but not included in e_i, and C_i containing e_i and all hyperedges included in e_i (see Fig. 1). Therefore we have:

1. $b_{i'} < a_i$ for all $e_{i'} \in A_i$,
2. $a_{i'} < a_i \leq b_{i'}$ for all $e_{i'} \in B_i$, and
3. $a_i \leq a_{i'} \leq b_{i'} \leq b_i$ for all $e_{i'} \in C_i$.

Clearly, for every $j \leq |C_i|$ we have $A[i, j] = |e_i|$ since by definition of A, e_i is included in the solution, and adding any other $j - 1$ sets from C_i to the solution does not increase the size of the union. In the remainder of the proof, when we refer to an optimal solution corresponding to $A[i', j']$ for some indices i' and j' we always mean a solution that uses the maximum number of sets in $C_{i'}$.

For any $t \geq 0$ and $j = t + |C_i|$, the optimal solution contains exactly t sets in $A_i \cup B_i$. Fix an optimal solution OPT_i corresponding to $A[i, j]$ and let e_{i^*} be the hyperedge with largest $b_{e_{i^*}}$ in OPT_i that does not belong to C_i. We show that

$$A[i, j] = A[i^*, j - |C_i \setminus C_{i^*}|] + |e_i \setminus e_{i^*}|. \tag{2}$$

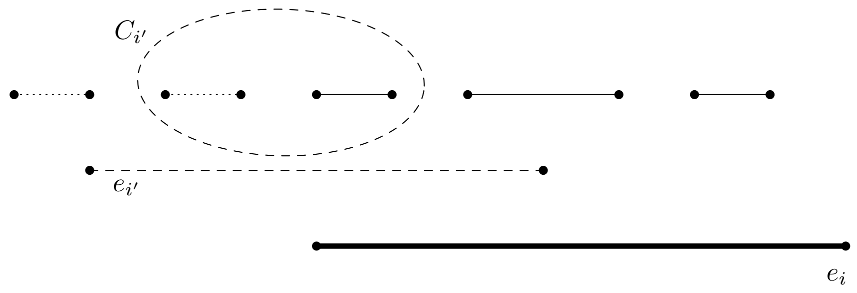


Figure 1 Partitioning of hyperedges induced by e_i. The dotted edges form set A_i, the dashed edge forms set B_i and the elements of C_i are represented by continuous edges. The set $C_{i'}$ is also shown in dashed pattern.

Then, by considering every hyperedge with index $i' < i$ as the possible i^* in Eq. (2) and taking the minimum value, one can compute $A[i, j]$ in linear time.

To complete the proof, we argue why Equation 2 holds. First observe that a solution with value $A[i, j]$ exists. Indeed, by adding all elements of $C_i \setminus C_{i^*}$ to an optimal solution for $A[i^*, j - |C_i \setminus C_{i^*}|]$ we obtain a solution for $A[i, j]$ covering exactly $|e_i \setminus e_{i^*}|$ additional elements. Next, assume that the value of $A[i, j]$ is less than that of Equation 2. Then we can obtain a solution for $A[i^*, j - |C_i \setminus C_{i^*}|]$ by removing from OPT_i all the elements in $|C_i \setminus C_{i^*}|$ to obtain a solution with value at most $A[i, j] - |e_i \setminus e_{i^*}|$, contradicting the fact that $A[i^*, j - |C_i \setminus C_{i^*}|]$ is the value of an optimal solution. ◀

8 Open problems

While no tight hardness results are known for Densest k-Subgraph and Smallest p-Edge Subgraph, there are lower bounds given by the log-density framework [7, 9]. In this framework, one considers the problem of distinguishing between a random graph and a graph which contains a planted dense subgraph. It has been conjectured that for certain parameters (namely, when the “log-density" of the subgraph is smaller than that of the host graph), this task is impossible, thus giving lower bounds on the approximability of these problems. In the graph setting, the existing algorithm of [7, 9] match these lower bounds.

However, in the hypergraph case, our current algorithms are still far from the corresponding lower bounds. In c-uniform hypergraphs, the lower bounds predicted by the log-density framework are $n^{(c-1)/4}$ for Densest k-Subhypergraph and $n^{1-2/(\sqrt{c}+1)}$ for Min p-Union. For $c = 3$, for example, these lower bounds give $n^{1/2}$ and $n^{2-\sqrt{3}} = n^{0.2679\cdots}$, respectively (contrast with our current guarantees of $n^{0.6978\cdots}$ and $n^{0.4}$). The existing approach for the graph case does not seem to easily carry over to hypergraphs, and it remains a technical challenge to match the log-density based predictions for hypergraphs of bounded rank.

For arbitrary rank, the lower bound given by the log-density framework is $m^{1/4}$ (note that we do not expect to achieve approximations that are sublinear in n in this case), as opposed to our current guarantee of $\sqrt{m}$. In general hypergraphs, one may also hope for hardness results which at the moment are elusive for the graph case or for bounded rank hypergraphs.

There is also an interesting connection between MpU/DkSH and the *Small-Set Vertex Expansion* problem (SSVE) [5, 20, 19]. In Small-Set Vertex Expansion we are given a graph

G and a parameter δ, and are asked to find the a set $V' \subseteq V$ with $|V'| \leq \delta n$ in order to minimize $\frac{|\{v \in V \setminus V' : v \in \Gamma(v)\}|}{|V'|}$. Given a graph G, consider the collection of neighborhoods $\hat{E} = \{\Gamma(v) : v \in V\}$ and the hypergraph $H = (V, \hat{E})$. If we let $p = \delta n$, the MpU problem (choosing p hyperedges in H to minimize their union) is quite similar to the SSVE problem. The main difference is that SSVE only "counts" nodes that are in $V \setminus V'$, while MpU would also count nodes in V'. It is known [21] that this special case of MpU reduces to SSVE, so it is no harder than SSVE, but it is not clear how much easier it is. This motivates the study of MpU when hyperedges are neighborhoods in an underlying graph, and studying the approximability of this problem is an interesting future direction.

References

1 Noga Alon, Sanjeev Arora, Rajsekar Manokaran, Dana Moshkovitz, and Omri Weinstein. Inapproximability of densest k-subgraph from average case hardness. Unpublished manuscript, 2011.

2 Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM Journal on Computing*, 42(5):2008–2037, 2013. doi:10.1137/120884857.

3 Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC'10, pages 171–180, 2010.

4 S. Arora, B. Barak, M. Brunnermeier, and R. Ge. Complicational complexity and information asymmetry in finnancial products. Submitted, 2016.

5 Sanjeev Arora and Rong Ge. *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*, chapter New Tools for Graph Coloring, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-22935-0_1.

6 Aditya Bhaskara. *Finding Dense Structures in Graphs and Matrices*. PhD thesis, Princeton University, 2012.

7 Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 201–210, 2010.

8 Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization, Third International Workshop, APPROX 2000, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, pages 84–95, 2000. doi:10.1007/3-540-44436-X_10.

9 Eden Chlamtac, Michael Dinitz, and Robert Krauthgamer. Everywhere-sparse spanners via dense subgraphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 758–767, 2012.

10 Julia Chuzhoy, Yury Makarychev, Aravindan Vijayaraghavan, and Yuan Zhou. Approximation algorithms and hardness of the k-route cut problem. *ACM Trans. Algorithms*, 12(1):2:1–2:40, December 2015. doi:10.1145/2644814.

11 Michael Dinitz, Guy Kortsarz, and Zeev Nutov. Improved Approximation Algorithm for Steiner k-Forest with Nearly Uniform Weights. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 115–127, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.

dagstuhl.de/opus/volltexte/2014/4692, doi:10.4230/LIPIcs.APPROX-RANDOM.2014.115.

12 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC'02, pages 534–543, New York, NY, USA, 2002. ACM. doi:10.1145/509907.509985.

13 Uriel Feige, Guy Kortsarz, and David Peleg. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

14 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485, 2012.

15 Anupam Gupta, Mohammadtaghi Hajiaghayi, Viswanath Nagarajan, and R. Ravi. Dial a ride from k-forest. *ACM Trans. Algorithms*, 6(2):41:1–41:21, April 2010. doi:10.1145/1721837.1721857.

16 Subhash Khot. Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006. doi:10.1137/S0097539705447037.

17 Christian Konrad and Adi Rosén. Approximating semi-matchings in streaming and in two-party communication. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 637–649, 2013.

18 Guy Kortsarz and David Peleg. On choosing a dense subgraph. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 692–701, 1993.

19 Anand Louis and Yury Makarychev. Approximation Algorithms for Hypergraph Small Set Expansion and Small Set Vertex Expansion. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 339–355, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2014/4707, doi:10.4230/LIPIcs.APPROX-RANDOM.2014.339.

20 Anand Louis, Prasad Raghavendra, and Santosh Vempala. The complexity of approximating vertex expansion. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 0:360–369, 2013. doi:10.1109/FOCS.2013.46.

21 Yuru Makarychev. Personal communication.

22 Zeev Nutov. Approximating steiner networks with node-weights. *SIAM Journal on Computing*, 39(7):3001–3022, 2010. doi:10.1137/080729645.

A Finding a Set of Minimum Expansion

Given a bipartite graph $G = (E, V, F)$, the subroutine $\textsc{Min-Exp}(G)$ returns a subset of E so that

$$\frac{|\textsc{Min-Exp}(G)|}{|\Gamma_G(\textsc{Min-Exp}(G))|} \geq \frac{|E'|}{|\Gamma_G(E')|},$$

for every subset $E' \subseteq E$. Minimally expanding subsets of this kind have previously been used (e.g. in [17, 14]) in communication settings where computation time is disregarded. We therefore present a polynomial time implementation for $\textsc{Min-Exp}$ using network flows. An alternative algorithm can be derived from a straightforward adaptation of a linear

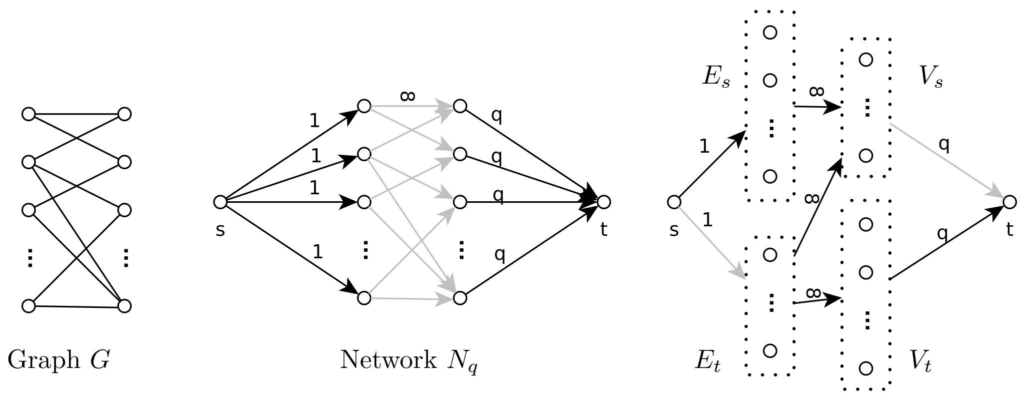


Figure 2 Left: Input graph G. Center: Network N_q. Right: Min-s-t-cut. Gray edges are cut edges.

programming approach for the graph case due to Charikar [8] to our setting (see Appendix B for more details).

Let $N_q = (\tilde{G}, c_q, s, t)$ be a flow network with directed bipartite graph $\tilde{G} = (E \cup \{t\}, V \cup \{s\}, \tilde{F})$, capacities c_q parameterized by a parameter q with $\frac{m}{n} < q < m$, source s and sink t as follows (and as illustrated in Figure 2):

1. Vertex s is connected to every $e \in E$ via directed edges (leaving s) with capacity 1.
2. Every $v \in V$ is connected to t via a directed edge (directed towards t) with capacity q.
3. Edges from F are included in $\tilde{F}$ and directed from E-vertex to V-vertex with capacity ∞.

Denote by C^* a minimum s-t cut in N_q and let $val(C^*)$ be the value of the cut. Since cutting all edges incident to vertex s results in a cut of value m, the min-cut value is at most m and thus finite, and, in particular, no edge connecting E to V is included in the min-cut. Denote by E_s the set of E-vertices that, when removing the cut-edges from the graph, are incident to s, and let $E_t = E \setminus E_s$. Let $V_s = \Gamma_G(E_s)$ and let $V_t = V \setminus V_s$. Since removing C^* from $\tilde{G}$ separates s from t, all outgoing edges from V_s are included in C^*. Furthermore, since C^* is a minimum cut, none of the edges leaving V_t are contained in the cut. The resulting structure is illustrated on the right in Figure 2. The value of the cut is computed as follows:

$$val(F^*) = |E_t| + q \cdot |V_s|. \tag{3}$$

We prove now a property connecting the value of a minimum cut to the expansion of a subset of E. This property allows us then to define an efficient algorithm for Min-Exp.

▶ **Lemma 24.** *Let q be such that $\frac{m}{n} < q < m$. Then:*

$$val(F^*) < m \Leftrightarrow \exists E' \subseteq E : \frac{|E'|}{|\Gamma_G(E')|} > q.$$

Proof. Suppose that $val(F^*) < m$. We prove that $E' = E_s$ fulfills the claimed property. The value of the cut $val(F^*)$ is computed according to Inequality 3 as follows:

$$m > val(F^*) = |E_t| + q \cdot |V_s| = m - |E_s| + q \cdot |V_s| = m - |E'| + q \cdot |\Gamma_G(E')|,$$

which implies $\frac{|E'|}{|\Gamma_G(E')|} > q$ as desired.

Suppose now that there is a $E' \subseteq E$ such that $\frac{|E'|}{|\Gamma_G(E')|} > q$. Then the set of edges C consisting of those that connect s to $E \setminus E'$ and those that connect $\Gamma_G(E')$ to t form a cut. We compute $val(C)$:

$$val(C) \;=\; |E \setminus E'| + q|\Gamma_G(E')| = m - |E'| + q|\Gamma_G(E')| < m - |E'| + |E'| = m.$$

The fact that $val(C^*) \leq val(C)$ completes the proof. ◀

Lemma 24 allows us to test whether there is a subset $E' \subseteq E$ such that $\frac{|E'|}{|\Gamma_G(E')|} > q$, for some value of q. For every set $E' \subseteq E$, we have $\frac{|E'|}{|\Gamma_G(E')|} \in \{\frac{a}{b} : a \in \{1, \ldots, m\}, b \in \{1, \ldots, n\}\}$. We could thus test all values $\frac{a}{b} - \epsilon$, for $a \in \{1, \ldots, m\}, b \in \{1, \ldots, n\}$ and a small enough ϵ, in order to identify the desired set (or use a binary search to speed up the process). Since computing a min-cut can be done in polynomial time, we obtain the following theorem:

▶ **Theorem 25.** *Algorithm* Min-Exp *can be implemented in polynomial time.*

B An LP-based algorithm for Minimum Expansion

We use hypergraph notation in this section. So the goal is to find a set $E' \subseteq E$ which minimizes $|\cup_{e\in E'} e|/|E'|$ over all choices of E' (so there is no requirement that $|E'| = p$).

We use the following LP relaxation, which is a straightforward adaptation of Charikar's [8] algorithm for graphs.

$$\begin{aligned} \text{LP} = \min \quad & \sum_{i\in V} x_i & \\ \text{s.t.} \quad & \sum_{e\in E} y_e = 1 & \\ & x_i \geq y_e & \forall e \in E, i \in e \\ & x_i \geq 0 & \forall i \in V \\ & y_e \geq 0 & \forall e \in E \end{aligned}$$

Consider the following simple rounding algorithm:

- Pick $r \in_R [0, 1]$ uniformly at random.
- Let $E' = \{e \in E \mid x_e \geq r\}$.
- Let $V' = \bigcup_{e\in E'} e$.

Clearly, for every vertex $e \in E$ we have

$$\text{Prob}[e \in E'] = y_e.$$

Also, for every vertex $i \in V$ we have

$$\text{Prob}[i \in V'] = \max_{e \ni i} y_e \leq x_i.$$

Therefore, by linearity of expectation, we have

$$\mathbb{E}[\text{LP} \cdot |E'| - |V'|] \geq \text{LP} \cdot 1 - \text{LP} = 0,$$

and this is obviously still true when we condition the expectation on $|E'| > 0$ (a positive probability event), so with positive probability, we get a pair (V_0, E_0) such that $E_0 \neq \emptyset$, $V_0 = \bigcup_{e\in E_0} e$ and $|V_0|/|E_0| \leq \text{LP}$. The rounding is trivially derandomized by trying $r = y_e$ for every vertex $e \in E$.

Online Row Sampling

Michael B. Cohen[*1], Cameron Musco[†2], and Jakub Pachocki[‡3]

1 Massachusetts Institute of Technology, Cambridge, MA, USA
micohen@mit.edu

2 Massachusetts Institute of Technology, Cambridge, MA, USA
cnmusco@mit.edu

3 Carnegie Mellon University, Pittsburgh, PA, USA
pachocki@cs.cmu.edu

Abstract

Finding a small spectral approximation for a tall $n \times d$ matrix $\mathbf{A}$ is a fundamental numerical primitive. For a number of reasons, one often seeks an approximation whose rows are sampled from those of $\mathbf{A}$. Row sampling improves interpretability, saves space when $\mathbf{A}$ is sparse, and preserves row structure, which is especially important, for example, when $\mathbf{A}$ represents a graph.

However, correctly sampling rows from $\mathbf{A}$ can be costly when the matrix is large and cannot be stored and processed in memory. Hence, a number of recent publications focus on row sampling in the streaming setting, using little more space than what is required to store the outputted approximation [12, 11].

Inspired by a growing body of work on online algorithms for machine learning and data analysis, we extend this work to a more restrictive *online* setting: we read rows of $\mathbf{A}$ one by one and immediately decide whether each row should be kept in the spectral approximation or discarded, without ever retracting these decisions. We present an extremely simple algorithm that approximates $\mathbf{A}$ up to multiplicative error ϵ and additive error δ using $\mathcal{O}(d \log d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ online samples, with memory overhead proportional to the cost of storing the spectral approximation. We also present an algorithm that uses $\mathcal{O}(d^2)$ memory but only requires $\mathcal{O}(d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ samples, which we show is optimal.

Our methods are clean and intuitive, allow for lower memory usage than prior work, and expose new theoretical properties of leverage score based matrix approximation.

1998 ACM Subject Classification F.2.1 [Numerical Algorithms and Problems] Computations on Matrices, F.1.2 [Modes of Computation] Online computation

Keywords and phrases spectral sparsification, leverage score sampling, online sparsification

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.7

1 Introduction

1.1 Background

A spectral approximation to a tall $n \times d$ matrix $\mathbf{A}$ is a smaller, typically $\tilde{\mathcal{O}}(d) \times d$ matrix $\tilde{\mathbf{A}}$ such that $\|\tilde{\mathbf{A}}\mathbf{x}\|_2 \approx \|\mathbf{A}\mathbf{x}\|_2$ for all $\mathbf{x}$. Typically one asks for a multiplicative approximation, which guarantees that $(1-\epsilon)\|\mathbf{A}\mathbf{x}\|_2^2 \leq \|\tilde{\mathbf{A}}\mathbf{x}\|_2^2 \leq (1+\epsilon)\|\mathbf{A}\mathbf{x}\|_2^2$. In other notation,

$$(1-\epsilon)\mathbf{A} \preceq \tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}.$$

[*] Michael B. Cohen is partially supported by NSF grant CCF-1111109.
[†] Cameron Musco is partially supported by NSF Graduate Research Fellowship No. 1122374, AFOSR grant FA9550-13-1-0042 and the NSF Center for Science of Information.
[‡] Jakub Pachocki is partially supported by NSF grant CCF-1065106.

Such approximations have many applications, most notably for solving least squares regression over $\mathbf{A}$ [6, 8]. If $\mathbf{A}$ is the vertex edge incidence matrix of a graph, $\tilde{\mathbf{A}}$ is a *spectral sparsifier* [20]. It can be used to approximate effective resistances, spectral clustering, mixing time and random walk properties, and many other computations.

A number of recent papers focus on fast algorithms for spectral approximation. Using sparse random subspace embeddings [6, 18, 17], it is possible to find $\tilde{\mathbf{A}}$ in input sparsity time, essentially by randomly recombining the rows of $\mathbf{A}$ into a smaller number of rows. In some cases these embeddings are not enough, as it is desirable for the rows of $\tilde{\mathbf{A}}$ to be a subset of rows sampled from $\mathbf{A}$. If $\mathbf{A}$ is sparse, this ensures that $\tilde{\mathbf{A}}$ is also sparse. If $\mathbf{A}$ represents a graph, it ensures that $\tilde{\mathbf{A}}$ is also a graph, specifically a weighted subgraph of the original.

It is well known that sampling $\mathcal{O}(d \log d/\epsilon^2)$ rows of $\mathbf{A}$ with probabilities proportional to their *leverage scores* yields a $(1+\epsilon)$ multiplicative factor spectral approximation to $\mathbf{A}$. Further, this sampling can be done in input sparsity time, either using subspace embeddings to approximate leverage scores, or using iterative sampling techniques [15], some that only work with subsampled versions of the original matrix [8].

1.2 Streaming and Online Row Sampling

When $\mathbf{A}$ is very large, input sparsity runtimes are not enough – memory restrictions also become important. Hence, recent work has tackled row sampling in a streaming model of computation. [12] presents a simple algorithm for sampling rows from an insertion only stream, using space approximately proportional to the size of the final approximation. [11] gives a sparse-recovery based algorithm that works in dynamic streams with row insertions and deletions, also using nearly optimal space. Unfortunately, to handle dynamic streams, the algorithm in [11] is complex, requires additional restrictions on the input matrix, and uses significantly suboptimal runtime to recover a spectral approximation from its low memory representation of the input stream.

While the algorithm in [12] is simple and efficient, we believe that its proof is incomplete, and do not see an obvious way to fix it. The main idea behind the algorithm is to sample rows by their leverage scores with respect to the stream seen so far. These leverage scores may be coarse overestimates of the true scores. However as more rows are streamed in, better estimates can be obtained and the sampled rows pruned to a smaller set. Unfortunately, the probability of sampling a row becomes dependent on which other rows are sampled. This seems to break the argument in that paper, which essentially claims that their process has the same distribution as would a single round of leverage score sampling.

In this paper we initiate the study of row sampling in an *online setting.* As in an insertion stream, we read rows of $\mathbf{A}$ one by one. However, upon seeing a row, we immediately decide whether it should be kept in the spectral approximation or discarded, without ever retracting these decisions. We present a similar algorithm to [12], however, since we never prune previously sampled rows, the probability of sampling a row only depends on whether previous rows in the stream were sampled. This limited dependency structure allows us to rigorously argue that a spectral approximation is obtained.

In addition to addressing gaps in the literature on streaming spectral approximation, our restricted model extends work on online algorithms for a variety of other machine learning and data analysis problems, including principal component analysis [4], clustering [16], classification [3, 10], and regression [10]. In practice, online algorithms are beneficial since they can be highly computationally and memory efficient. Further, they can be applied in scenarios in which data is produced in a continuous stream and intermediate results must be output as the stream is processed. Spectral approximation is a widely applicable primitive for approximate learning and computation, so studying its implementation in an online setting is a natural direction.

1.3 Our Results

Our primary contribution is a very simple algorithm for leverage score sampling in an online manner. The main difficultly with row sampling using leverage scores is that leverage scores themselves are not easy to compute. They are given by $l_i = \mathbf{a}_i^T(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{a}_i$, and so require solving systems in $\mathbf{A}^T\mathbf{A}$ if computed naively. This is not only expensive, but also impossible in an online setting, where we do not have access to all of $\mathbf{A}$.

A critical observation is that it always suffices to sample rows by overestimates of their true leverage scores. The number of rows that must be sampled is proportional to the sum of these overestimates. Since the leverage score of a row can only go up when we remove rows from the matrix, a simple way to obtain an overestimate is to compute leverage score using just a subset of the other rows of $\mathbf{A}$. That is, letting $\mathbf{A}_j$ contain just j of $\mathbf{A}$'s n rows, we can overestimate l_i by $\tilde{l}_i = \mathbf{a}_i^T(\mathbf{A}_j^T\mathbf{A}_j)^{-1}\mathbf{a}_i$

[8] shows that if $\mathbf{A}_j$ is a subset of rows sampled uniformly at random, then the expected leverage score of $\mathbf{a}_i$ is d/j. This simple fact immediately gives a result for online sampling from a *randomly ordered stream*. If we compute the leverage score of the current row $\mathbf{a}_i$ against all previously seen rows (or some approximation to these rows), then the expected sum of our overestimates is bounded by $d + d/2 + ... + ... + d/n = \mathcal{O}(d \log n)$. So, sampling $\mathcal{O}(d \log d \log n/\epsilon^2)$ rows is enough obtain a $(1+\epsilon)$ multiplicative factor spectral approximation.

What if we cannot guarantee a randomly ordered input stream? Is there any hope of being able to compute good leverage score estimates in an online manner? Surprisingly the answer to this is yes - we can in fact run nearly the exact same algorithm and be guaranteed that the sum of estimated leverage scores is low, *regardless of stream order*. Roughly, each time we receive a row which has high leverage score with respect to the previous rows, it must compose a significant part of $\mathbf{A}$'s spectrum. If $\mathbf{A}$ does not continue to grow unboundedly, there simply cannot be too many of these significant rows.

Specifically, we show that if we sample by the *ridge leverage scores* [1] over all previously seen rows, which are the leverage scores computed over $\mathbf{A}_i^T\mathbf{A}_i + \lambda\mathbf{I}$ for some small regularizing factor λ, then with just $\mathcal{O}(d \log d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ samples we obtain a $(1+\epsilon)$ multiplicative, δ additive error spectral approximation. That is, with high probability we sample a matrix $\tilde{\mathbf{A}}$ with $(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I}$.

To gain intuition behind this bound, note that we can convert it into a multiplicative one by setting $\delta = \epsilon\sigma_{min}(\mathbf{A})^2$ (as long as we have some estimate of $\sigma_{min}(\mathbf{A})$). This setting of δ will require taking $\mathcal{O}(d \log d \log(\kappa(\mathbf{A}))/\epsilon^2)$ samples. If we have a polynomial bound on the condition number of $\mathbf{A}$, as we do, for instance, for graphs with polynomially bounded edges weights, this becomes $\mathcal{O}(d \log^2 d/\epsilon^2)$ – nearly matching the $\mathcal{O}(d \log d/\epsilon^2)$ achievable if sampling by true leverage scores.

Our online sampling algorithm is extremely simple. When each row comes in, we compute the online ridge leverage score, or an estimate of it, and then irrevocably either add the row to our approximation or remove it. As mentioned, it is similar in form to the streaming algorithm of [12], except that it does not require pruning previously sampled rows. This allows us to avoid difficult dependency issues. Additionally, without pruning, we do not even need to store all previously sampled rows. As long as we store a constant factor spectral approximation our previous samples, we can compute good approximations to the online ridge leverage scores. In this way, we can store just $\mathcal{O}(d \log d \log(\epsilon\|\mathbf{A}\|_2^2/\delta))$ rows in working memory ($\mathcal{O}(d \log^2 d)$ if we want a spectral graph sparsifier), filtering our input stream into an $\mathcal{O}(d \log d \log(\kappa(\mathbf{A}))/\epsilon^2)$ sized output stream. Note that this memory bound in fact *improves* as ϵ decreases, and regardless, can be significantly smaller than the output size of the algorithm.

In addition to our main sampling result, we use our bounds on online ridge leverage score approximations to show that an algorithm in the style of [2] allows us to remove a $\log d$ factor and sample just $\mathcal{O}(d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ rows (Theorem 10). This algorithm is more complex and can require $\mathcal{O}(d^2)$ working memory. However, in Theorem 12 we show that it is asymptotically optimal. The $\log(\epsilon\|\mathbf{A}\|_2^2/\delta)$ factor is not an artifact of our analysis, but is truly the cost of the restricting ourselves to online sampling. No algorithm can obtain a multiplicative $(1+\epsilon)$ additive δ spectral approximation taking fewer than $\Omega(d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ rows in an online manner.

2 Overview

Let $\mathbf{A}$ be an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$. A natural approach to row sampling from $\mathbf{A}$ is picking an *a priori* probability with which each row is kept, and then deciding whether to keep each row independently. A common choice is for the sampling probabilities to be proportional to the *leverage scores* of the rows. The leverage score of the i-th row of $\mathbf{A}$ is defined to be

$$\mathbf{a}_i^T(\mathbf{A}^T\mathbf{A})^\dagger\mathbf{a}_i,$$

where the dagger symbol denotes the pseudoinverse. In this work, we will be interested in approximating $\mathbf{A}^T\mathbf{A}$ with some (very) small multiple of the identity added. Hence, we will be interested in the λ*-ridge leverage scores* [1]:

$$\mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i,$$

for a parameter $\lambda > 0$.

In many applications, obtaining the (nearly) exact values of $\mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i$ for sampling is difficult or outright impossible. A key idea is that as long as we have a sequence $l_1, \ldots, l_n$ of *overestimates* of the λ-ridge leverage scores, that is for $i = 1, \ldots, n$

$$l_i \geq \mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i,$$

we can sample by these overestimates and obtain rigorous guarantees on the quality of the obtained spectral approximation. This notion is formalized in Theorem 1.

▶ **Theorem 1.** *Let $\mathbf{A}$ be an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$. Let $\epsilon \in (0,1), \delta > 0, \lambda := \delta/\epsilon, c := 8 \log d/\epsilon^2$. Assume we are given $l_1, \ldots, l_n$ such that for all $i = 1, \ldots, n$,*

$$l_i \geq \mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i.$$

For $i = 1, \ldots, n$, let $p_i := \min(cl_i, 1)$. Construct $\tilde{\mathbf{A}}$ by independently sampling each row $\mathbf{a}_i$ of $\mathbf{A}$ with probability p_i, and rescaling it by $1/\sqrt{p_i}$ if it is included in the sample. Then, with high probability,

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I},$$

and the number of rows in $\tilde{\mathbf{A}}$ is $\mathcal{O}\left(\left(\sum_{i=1}^n l_i\right) \log d/\epsilon^2\right)$.

Proof. This sort of guarantee for leverage score sampling is well known. See for example Lemma 4 of [8]. If we sampled both the rows of $\mathbf{A}$ and the rows of $\sqrt{\lambda}\mathbf{I}$ with the leverage scores over $(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})$, we would have $(1-\epsilon)(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}) \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})$. However, we do not sample the rows of the identity. Since we could have sampled them each with probability 1, we can simply subtract $\lambda\mathbf{I} = (\delta/\epsilon)\mathbf{I}$ from the multiplicative bound and have: $(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I}$. ◀

The idea of using overestimates of leverage scores to perform row sampling has been applied successfully to various problems (see e.g. [13, 8]). However, in these applications, access to the entire matrix is required beforehand. In the streaming and online settings, we have to rely on partial data to approximate the true leverage scores. The most natural idea is to just use the portion of the matrix seen thus far as an approximation to $\mathbf{A}$. This leads us to introduce the *online λ-ridge leverage scores*:

$$l_i := \min(\mathbf{a}_i^T(\mathbf{A}_{i-1}^T\mathbf{A}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i, 1),$$

where $\mathbf{A}_i$ $(i = 0, \ldots, n)$ is defined as the matrix consisting of the first i rows of $\mathbf{A}$[1].

Since clearly $\mathbf{A}_i^T\mathbf{A}_i \preceq \mathbf{A}^T\mathbf{A}$ for all i, it is not hard to see that l_i does overestimate the true λ-ridge leverage score for row $\mathbf{a}_i$. A more complex question, however, is establishing an upper bound on $\sum_{i=1}^n l_i$ so that we can bound the number of samples needed by Theorem 1.

A core result of this work, stated in Theorem 2, is establishing such an upper bound; in fact, this bound is shown to be tight up to constants (Theorem 12) and is nearly-linear in most cases.

▶ **Theorem 2.** *Let $\mathbf{A}$ be an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$. Let $\mathbf{A}_i$ for $i \in \{0, \ldots, n\}$ be the matrix consisting of the first i rows of $\mathbf{A}$. For $\lambda > 0$, let*

$$l_i := \min(\mathbf{a}_i^T(\mathbf{A}_{i-1}^T\mathbf{A}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i, 1).$$

be the online λ-ridge leverage score of the i^{th} row of $\mathbf{A}$. Then

$$\sum_{i=1}^n l_i = \mathcal{O}(d \log(\|\mathbf{A}\|_2^2/\lambda)).$$

Theorems 2 and 1 suggest a simple algorithm for online row sampling: simply use the online λ-ridge leverage scores, for $\lambda := \delta/\epsilon$. This produces a spectral approximation with only $\mathcal{O}(d \log d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ rows. Unfortunately, computing l_i exactly requires us to store *all* the rows we have seen in memory (or alternatively to store the sum of their outer products, $\mathbf{A}_i^T\mathbf{A}_i$). In many cases, such a requirement would defeat the purpose of streaming row sampling.

A natural idea is to use the sample we have kept thus far as an approximation to $\mathbf{A}_i$ when computing l_i. It turns out that the approximate online ridge leverage scores $\tilde{l}_i$ computed in this way will not always be good approximations to l_i; however, we can still prove that they satisfy the requisite bounds and yield the same row sample size! We formalize these results in the algorithm Online-Sample (Figure 1) and Theorem 3.

▶ **Theorem 3.** *Let $\tilde{\mathbf{A}}$ be the matrix returned by* Online-Sample$(\mathbf{A}, \epsilon, \delta)$. *With high probability,*

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I},$$

and the number of rows in $\tilde{\mathbf{A}}$ is $\mathcal{O}(d \log d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$.

[1] We use the proposed scores l_i for simplicity, however note that the following, perhaps more natural, definition of online leverage scores would also be effective: $l_i' := \mathbf{a}_i^T(\mathbf{A}_i^T\mathbf{A}_i + \lambda\mathbf{I})^{-1}\mathbf{a}_i$.

```
$\tilde{\mathbf{A}} = $ ONLINE-SAMPLE$(\mathbf{A}, \epsilon, \delta)$, where $\mathbf{A}$ is an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$,
$\epsilon \in (0,1)$, $\delta > 0$.
1. Set $\lambda := \delta/\epsilon$, $c := 8 \log d/\epsilon^2$.
2. Let $\tilde{\mathbf{A}}_0$ be a $0 \times d$ matrix.
3. For $i = 1, \ldots, n$:
   a. Let $\tilde{l}_i := \min((1+\epsilon)\mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i, 1)$.
   b. Let $p_i := \min(c\tilde{l}_i, 1)$.
   c. Set $\tilde{\mathbf{A}}_i := \begin{cases} \begin{bmatrix} \tilde{\mathbf{A}}_{i-1} \\ \mathbf{a}_i/\sqrt{p_i} \end{bmatrix} & \text{with probability } p_i, \\ \tilde{\mathbf{A}}_{i-1} & \text{otherwise.} \end{cases}$
4. Return $\tilde{\mathbf{A}} := \tilde{\mathbf{A}}_n$.
```

Figure 1 The basic online sampling algorithm.

To save computation, we note that, with a small modification to our analysis, we can run ONLINE-SAMPLE with batch processing of rows. Specifically, say we start from the i^{th} position in the stream. we can store the next $b = \mathcal{O}(d)$ rows. We can then compute sampling probabilities for these rows all at once using a system solver for $(\tilde{\mathbf{A}}_{i+b}^T\tilde{\mathbf{A}}_{i+b} + \lambda\mathbf{I})$. Using a trick introduced in [19], by applying a Johnson-Lindenstrauss random projection to the rows whose scores we are computing, we need just $\mathcal{O}(\log(1/\delta))$ system solves to compute constant factor approximations to the ridge scores with probability $1-\delta$. If we set $\delta = 1/\text{poly}(n)$ then we can union bound over our whole stream, using this trick with each batch of $\mathcal{O}(d)$ input rows. The batch probabilities will only be closer to the true ridge leverage scores than the non-batch probabilities and we will enjoy the same guarantees as ONLINE-SAMPLE.

Additionally, it turns out that with a simple trick, it is possible to reduce the memory usage of the algorithm by a factor of ϵ^{-2}, bringing it down to $\mathcal{O}(d \log d \log(\epsilon\|A\|_2^2/\delta))$ (assuming the row sample is output to an output stream). Note that this expression gets *smaller* with ϵ; hence we obtain a row sampling algorithm with memory complexity independent of desired multiplicative precision. The basic idea is that, instead of keeping all previously sampled rows in memory, we store a smaller set of rows that give a constant factor spectral approximation, still enough to give good estimates of the online ridge leverage scores.

This result is presented in the algorithm SLIM-SAMPLE (Figure 2) and Lemma 9. A particularly interesting consequence for graphs with polynomially bounded edge weights is:

▶ **Corollary 4.** *Let G be a simple graph on d vertices, and $\epsilon \in (0,1)$. We can construct a $(1+\epsilon)$-sparsifier of G of size $\mathcal{O}(d \log^2 d/\epsilon^2)$, using only $\mathcal{O}(d \log^2 d)$ working memory in the online model.*

Proof. This follows simply from applying Theorem 3 with $\delta = \epsilon/\sigma_{min}^2(\mathbf{A})$. For an unweighted graph on d vertices, $\|\mathbf{A}\|_2^2 \leq d$, since d is the largest squared singular value of the complete graph. Combining with Lemma 6.1 of [21], we have that the condition number of a graph on d vertices whose edge weights are within a multiplicative poly(d) of each other is polynomial in d. So $\log(\epsilon\|\mathbf{A}\|_2^2/\delta) = \log(\kappa^2(\mathbf{A})) = \mathcal{O}(\log d)$. ◀

We remark that the algorithm of Corollary 4 can be made to run in nearly linear time in the stream size. We combine SLIM-SAMPLE with the batch processing idea described above. Because $\mathbf{A}$ is a graph, our matrix approximation is always a symmetric diagonally dominant matrix, with $\mathcal{O}(d)$ nonzero entries. We can solve systems in it in time $\tilde{\mathcal{O}}(d)$. Using the

Johnson-Lindenstrauss random projection trick of [19], we can compute approximate ridge leverage scores for a batch of $\mathcal{O}(d)$ rows with failure probability polynomially small in n in $\tilde{\mathcal{O}}(d \log n)$ time. Union bounding over the whole stream, we obtain nearly linear runtime.

To complement the row sampling results discussed above, we explore the limits of the proposed online setting. In Section 4 we present the algorithm Online-BSS, which obtains spectral approximations with $\mathcal{O}(d \log(\epsilon \|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ rows in the online setting (with larger memory requirements than the simpler sampling algorithms). Its analysis is given in Theorem 10. In Section 5, we show that this number of samples is in fact the best achievable, up to constant factors (Theorem 12). The $\log(\epsilon \|\mathbf{A}\|_2^2/\delta)$ factor is truly the cost of requiring rows to be selected in an online manner.

3 Analysis of Sampling Schemes

We begin by bounding the sum of online λ-ridge leverage scores. The intuition behind the proof of Theorem 2 is that whenever we add a row with a large online leverage score to a matrix, we increase its determinant significantly, as follows from the matrix determinant lemma (Lemma 5). Thus we can reduce upper bounding the online leverage scores to bounding the matrix determinant.

▶ **Lemma 5** (Matrix determinant lemma). *Assume* $\mathbf{S}$ *is an invertible square matrix and* $\mathbf{u}$ *is a vector. Then*

$$\det(\mathbf{S} + \mathbf{u}\mathbf{u}^T) = (\det \mathbf{S})(1 + \mathbf{u}^T \mathbf{S}^{-1} \mathbf{u}).$$

Proof of Theorem 2. By Lemma 5, we have

$$\begin{aligned}\det(\mathbf{A}_{i+1}^T \mathbf{A}_{i+1} + \lambda \mathbf{I}) &= \det(\mathbf{A}_i^T \mathbf{A}_i + \lambda \mathbf{I}) \cdot \left(1 + \mathbf{a}_{i+1}^T (\mathbf{A}_i^T \mathbf{A}_i + \lambda \mathbf{I})^{-1} \mathbf{a}_{i+1}\right) \\ &\geq \det(\mathbf{A}_i^T \mathbf{A}_i + \lambda \mathbf{I}) \cdot (1 + l_{i+1}) \\ &\geq \det(\mathbf{A}_i^T \mathbf{A}_i + \lambda \mathbf{I}) \cdot e^{l_{i+1}/2}.\end{aligned}$$

Hence,

$$\begin{aligned}\det(\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I}) &= \det(\mathbf{A}_n^T \mathbf{A}_n + \lambda \mathbf{I}) \\ &\geq \det(\lambda \mathbf{I}) \cdot e^{\sum l_i/2} \\ &= \lambda^d e^{\sum l_i/2}.\end{aligned}$$

We have $\det(\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I}) \leq (\|\mathbf{A}\|_2^2 + \lambda)^d$. Therefore

$$(\|\mathbf{A}\|_2^2 + \lambda)^d \geq \lambda^d e^{\sum l_i/2}.$$

Taking logarithms of both sides, we obtain

$$\begin{aligned}d \log(\|\mathbf{A}\|_2^2 + \lambda) &\geq d \log \lambda + \sum l_i/2 \\ \sum l_i &\leq 2d \log(1 + \|\mathbf{A}\|_2^2/\lambda).\end{aligned}$$ ◀

We now turn to analyzing the algorithm Online-Sample. Because the samples taken by the algorithm are *not* independent, we are not able to use a standard matrix Chernoff bound like the one in Theorem 1. However, we do know that whether we take row i does not depend on later rows; thus, we are able to analyze the process as a martingale. We will use a matrix version of the Freedman inequality given by Tropp.

▶ **Theorem 6** (Matrix Freedman inequality [22]). *Let $\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_n$ be a matrix martingale whose values are self-adjoint matrices with dimension d, and let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be the difference sequence. Assume that the difference sequence is uniformly bounded in the sense that*

$$\|\mathbf{X}_k\|_2 \le R \text{ almost surely, for } k = 1, \ldots, n.$$

Define the predictable quadratic variation process of the martingale:

$$\mathbf{W}_k := \sum_{j=1}^{k} \mathbf{E}_{j-1}\left[\mathbf{X}_j^2\right], \text{ for } k = 1, \ldots, n.$$

Then, for all $\epsilon > 0$ and $\sigma^2 > 0$,

$$\mathbf{P}\left[\|\mathbf{Y}_n\|_2 \ge \epsilon \text{ and } \|\mathbf{W}_n\|_2 \le \sigma^2\right] \le d \cdot \exp\left(-\frac{-\epsilon^2/2}{\sigma^2 + R\epsilon/3}\right)$$

We begin by showing that the output of Online-Sample is in fact an approximation of $\mathbf{A}$, and that the approximate online leverage scores are lower bounded by the actual online leverage scores.

▶ **Lemma 7.** *After running* Online-Sample*, it holds with high probability that*

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I},$$

and also

$$\tilde{l}_i \ge \mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i$$

for $i = 1, \ldots, n$.

Proof. Let

$$\mathbf{u}_i := (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1/2}\mathbf{a}_i.$$

We construct a matrix martingale $\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_n \in \mathbb{R}^{d\times d}$ with the difference sequence $\mathbf{X}_1, \ldots, \mathbf{X}_n$. Set $\mathbf{Y}_0 = \mathbf{0}$. If $\|\mathbf{Y}_{i-1}\|_2 \ge \epsilon$, we set $\mathbf{X}_i := \mathbf{0}$. Otherwise, let

$$\mathbf{X}_i := \begin{cases} (1/p_i - 1)\mathbf{u}_i\mathbf{u}_i^T & \text{if } \mathbf{a}_i \text{ is sampled in } \tilde{\mathbf{A}}, \\ -\mathbf{u}_i\mathbf{u}_i^T & \text{otherwise.} \end{cases}$$

In the case that $\|\mathbf{Y}_{i-1}\|_2 < \epsilon$, by construction, $\|\mathbf{Y}_j\|_2 < \epsilon$ for all $j < i-1$. So we have:

$$\mathbf{Y}_{i-1} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1/2}(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} - \mathbf{A}_{i-1}^T\mathbf{A}_{i-1})(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1/2}.$$

Hence, we have

$$\begin{aligned} \tilde{l}_i &= \min((1+\epsilon)\mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i, 1) \\ &\ge \min((1+\epsilon)\mathbf{a}_i^T(\mathbf{A}_{i-1}^T\mathbf{A}_{i-1} + \lambda\mathbf{I} + \epsilon(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}))^{-1}\mathbf{a}_i, 1) \\ &\ge \min((1+\epsilon)\mathbf{a}_i^T((1+\epsilon)(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}))^{-1}\mathbf{a}_i, 1) \\ &= \mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i \qquad (1) \\ &= \mathbf{u}_i^T\mathbf{u}_i, \end{aligned}$$

and so $p_i \geq \min(c\mathbf{u}_i^T\mathbf{u}_i, 1)$. If $p_i = 1$, then $\mathbf{X}_i = 0$. Otherwise, we have $p_i \geq c\mathbf{u}_i^T\mathbf{u}_i$ and:

$$\|\mathbf{X}_i\|_2 \leq \max\{1, 1/p_i - 1\} \cdot \|\mathbf{u}_i\mathbf{u}_i^T\|_2 \leq \frac{1}{p_i}\mathbf{u}_i^T\mathbf{u}_i \leq 1/c. \tag{2}$$

Further

$$\begin{aligned}\mathbf{E}_{i-1}\left[\mathbf{X}_i^2\right] &\preceq p_i \cdot (1/p_i - 1)^2(\mathbf{u}_i\mathbf{u}_i^T)^2 + (1-p_i)\cdot(\mathbf{u}_i\mathbf{u}_i^T)^2\\ &= (\mathbf{u}_i\mathbf{u}_i^T)^2 \cdot (1-p_i)/p_i\\ &\preceq \mathbf{u}_i\mathbf{u}_i^T \cdot \left(\mathbf{u}_i^T\mathbf{u}_i/p_i\right)\\ &\preceq \mathbf{u}_i\mathbf{u}_i^T/c. && \text{(by equation (2))}\end{aligned}$$

And so, for the predictable quadratic variation process of the martingale $\{\mathbf{Y}_i\}$:

$$\mathbf{W}_i := \sum_{k=1}^{i} \mathbf{E}_{k-1}\left[\mathbf{X}_k^2\right],$$

we have

$$\|\mathbf{W}_i\|_2 \leq \left\|\sum_{k=1}^{i}\mathbf{u}_i\mathbf{u}_i^T/c\right\|_2 \leq 1/c.$$

Therefore by, Theorem 6, we have

$$\begin{aligned}\mathbf{P}\left[\|\mathbf{Y}_n\|_2 \geq \epsilon\right] &\leq d \cdot \exp\left(\frac{-\epsilon^2/2}{1/c + \epsilon/(3c)}\right)\\ &\leq d\cdot\exp(-c\epsilon^2/4)\\ &= 1/d.\end{aligned}$$

This implies that with high probability

$$\|(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1/2}(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I})(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1/2} - \mathbf{I}\|_2 \leq \epsilon$$

and so

$$(1-\epsilon)(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}) \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I} \preceq (1+\epsilon)(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}).$$

Subtracting $\lambda\mathbf{I} = (\delta/\epsilon)\mathbf{I}$ from all sides, we get

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I}.$$

Finally, note that, since we set $\mathbf{X}_i = \mathbf{0}$ if $\|\mathbf{Y}_{i-1}\|_2 \geq \epsilon$, $\|\mathbf{Y}_n\|_2 < \epsilon$ implies $\|\mathbf{Y}_i\|_2 < \epsilon$ for all $i < n$. We thus have the desired bound on $\tilde{l}_i$ by equation (1). ◀

If we set c in ONLINE-SAMPLE to be proportional to $\log n$ rather than $\log d$, we would be able to take a union bound over all the rows and guarantee that with high probability all the approximate online leverage scores $\tilde{l}_i$ are close to true online leverage scores l_i. Thus Theorem 2 would imply that ONLINE-SAMPLE only selects $\mathcal{O}(d\log n \log(\|\mathbf{A}\|_2^2/\lambda)/\epsilon^2)$ rows with high probability.

In order to remove the dependency on n, we have to sacrifice achieving close approximations to l_i at every step. Instead, we show that the *sum* of the computed approximate online leverage scores is still small with high probability, using a custom Chernoff bound.

▶ **Lemma 8.** *After running* Online-Sample*, it holds with high probability that*

$$\sum_{i=1}^{n} \tilde{l}_i = \mathcal{O}(d \log(\|\mathbf{A}\|_2^2/\lambda)).$$

Proof. Define

$$\delta_i := \log\det(\tilde{\mathbf{A}}_i^T \tilde{\mathbf{A}}_i + \lambda \mathbf{I}) - \log\det(\tilde{\mathbf{A}}_{i-1}^T \tilde{\mathbf{A}}_{i-1} + \lambda \mathbf{I}).$$

The proof closely follows the idea from the proof of Theorem 2. We will aim to show that large values of $\tilde{l}_i$ correlate with large values of δ_i. However, the sum of δ_i can be bounded by the logarithm of the ratio of the determinants of $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I}$ and $\lambda\mathbf{I}$. First, we will show that $\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right]$ is always at most 1. We begin by an application of Lemma 5.

$$\begin{aligned}\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right] &= p_i \cdot e^{\tilde{l}_i/8}(1 + \mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i/p_i)^{-1} + (1-p_i)e^{\tilde{l}_i/8} \\ &\le p_i \cdot (1 + \tilde{l}_i/4)(1 + \mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i/p_i)^{-1} + (1-p_i)(1+\tilde{l}_i/4).\end{aligned}$$

If $c\tilde{l}_i < 1$, we have $p_i = c\tilde{l}_i$ and $\tilde{l}_i = (1+\epsilon)\mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i$, and so:

$$\begin{aligned}\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right] &\le c\tilde{l}_i \cdot (1+\tilde{l}_i/4)(1 + 1/((1+\epsilon)c))^{-1} + (1 - c\tilde{l}_i)(1+\tilde{l}_i/4) \\ &= (1+\tilde{l}_i/4)(c\tilde{l}_i(1 + 1/((1+\epsilon)c))^{-1} + 1 - c\tilde{l}_i) \\ &\le (1+\tilde{l}_i/4)(1 + c\tilde{l}_i(1 - 1/(4c) - 1)) \\ &= (1+\tilde{l}_i/4)(1 - \tilde{l}_i/4) \\ &\le 1.\end{aligned}$$

Otherwise, we have $p_i = 1$ and so:

$$\begin{aligned}\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right] &\le (1+\tilde{l}_i/4)(1 + \mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i)^{-1} \\ &\le (1+\tilde{l}_i/4)(1+\tilde{l}_i)^{-1} \\ &\le 1.\end{aligned}$$

We will now analyze the expected product of $\exp(\tilde{l}_i/8 - \delta_i)$ over the first k steps. We group the expectation over the first k steps into one over the first $k-1$ steps, aggregating the expectation for the last step by using one-way independence. For $k \ge 1$ we have

$$\begin{aligned}\mathbf{E}\left[\exp\left(\sum_{i=1}^{k} \tilde{l}_i/8 - \delta_i\right)\right] &= \underset{\text{first } k-1 \text{ steps}}{\mathbf{E}}\left[\exp\left(\sum_{i=1}^{k-1} \tilde{l}_i/8 - \delta_i\right)\mathbf{E}_{k-1}\left[\exp(\tilde{l}_k/8 - \delta_k)\right]\right] \\ &\le \mathbf{E}\left[\exp\left(\sum_{i=1}^{k-1} \tilde{l}_i/8 - \delta_i\right)\right],\end{aligned}$$

and so by induction on k

$$\mathbf{E}\left[\exp\left(\sum_{i=1}^{n} \tilde{l}_i/8 - \delta_i\right)\right] \le 1.$$

Hence by Markov's inequality

$$\mathbf{P}\left[\sum_{i=1}^{n} \tilde{l}_i > 8d + 8\sum_{i=1}^{n} \delta_i\right] \le e^{-d}.$$

```
$\tilde{\mathbf{A}} = \textsc{Slim-Sample}(\mathbf{A}, \epsilon, \delta)$, where $\mathbf{A}$ is an $n \times d$ matrix with rows $\mathbf{a}_1, \dots, \mathbf{a}_n$,
$\epsilon \in (0,1)$, $\delta > 0$.
1. Set $\lambda := \delta/\epsilon$, $c := 8 \log d/\epsilon^2$.
2. Let $\tilde{\mathbf{A}}_0$ be a $0 \times d$ matrix.
3. Let $\tilde{l}_1, \dots, \tilde{l}_n$ be the approximate online leverage scores computed by an inde-
   pendent instance of $\textsc{Online-Sample}(\mathbf{A}, 1/2, \delta/(2\epsilon))$.
4. For $i = 1, \dots, n$:
   a. Let $p_i := \min(c\tilde{l}_i, 1)$.
   b. Set $\tilde{\mathbf{A}}_i := \begin{cases} \begin{bmatrix} \tilde{\mathbf{A}}_{i-1} \\ \mathbf{a}_i/\sqrt{p_i} \end{bmatrix} & \text{with probability } p_i, \\ \tilde{\mathbf{A}}_{i-1} & \text{otherwise.} \end{cases}$
5. Return $\tilde{\mathbf{A}} := \tilde{\mathbf{A}}_n$.
```

Figure 2 The low-memory online sampling algorithm.

By Lemma 7, with high probability we have $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I} \preceq (1+\epsilon)(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})$. We also have with high probability:

$$\det(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I}) \le (1+\epsilon)^d(\|\mathbf{A}\|_2^2 + \lambda)^d,$$
$$\log\det(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I}) \le d(1 + \log(\|\mathbf{A}\|_2^2 + \lambda)).$$

Hence, with high probability it holds that

$$\begin{aligned}\sum_{i=1}^n \delta_i &= \log\det(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I}) - d\log(\lambda) \\ &\le d(1 + \log(\|\mathbf{A}\|_2^2 + \lambda) - \log(\lambda)) \\ &= d(1 + \log(1 + \|\mathbf{A}\|_2^2/\lambda)).\end{aligned}$$

And so, with high probability,

$$\begin{aligned}\sum_{i=1}^n \tilde{l}_i &\le 8d + 8\sum_{i=1}^n \delta_i \\ &\le 9d + 8d\log(1 + \|\mathbf{A}\|_2^2/\lambda) \\ &= \mathcal{O}(d\log(\|\mathbf{A}\|_2^2/\lambda)).\end{aligned}$$ ◀

Proof of Theorem 3. The thesis follows immediately from Lemmas 7 and 8. ◀

We now consider a simple modification of Online-Sample that removes dependency on ϵ from the working memory usage with no additional cost.

▶ **Lemma 9.** *Let $\tilde{\mathbf{A}}$ be the matrix returned by* Slim-Sample$(\mathbf{A}, \epsilon, \delta)$. *Then, with high probability,*

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I},$$

and the number of rows in $\tilde{\mathbf{A}}$ is $\mathcal{O}(d\log d\log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$.

Moreover, with high probability the algorithm Slim-Sample*'s memory requirement is dominated by storing $\mathcal{O}(d\log d\log(\epsilon\|\mathbf{A}\|_2^2/\delta))$ rows of $\mathbf{A}$.*

Proof. As the samples are independent, the thesis follows from Theorem 1 and Lemmas 7 and 8. ◀

4 Asymptotically Optimal Algorithm

In addition to sampling by online leverage scores, there is also a variant of the "BSS" method [2] that applies in our setting. Like the original [2], this approach removes the $\log d$ factor from the row count of the output spectral approximation, matching the lower bound for online sampling given in Theorem 12.

Unlike [2] itself, our algorithm is randomized – it is similar to, and inspired by, the randomized version of BSS from [14], especially the simpler "Algorithm 1" from that paper (the main difference from that is considering each row separately). In fact, this algorithm is of the same form as the basic sampling algorithm, in that when each row comes in, a probability p_i is assigned to it, and it is kept (and rescaled) with probability p_i and rejected otherwise. The key difference is the definition of the p_i.

There are also some differences in the nature of the algorithm and its guarantees. Notably, the p_i cannot be computed solely based on the row sample output so far–it is necessary to "remember" the entire matrix given so far. This means that the BSS method is not memory efficient, using $O(d^2)$ space. Additionally, online leverage score sampling gives bounds on both the size of the output spectral approximation and its accuracy with high probability. In contrast, this method gives an *expected* bound on the output size, while it *never* fails to output a correct spectral approximation. Note that these guarantees are essentially the same as those in the appendix of [14].

One may, however, improve the memory dependence in some cases simply by running it on the output stream of the online leverage score sampling method. This reduces the storage cost to the size of that spectral approximation. The BSS method still does not produce an actual space *savings* (in particular, there is a still a $\log d$ factor in space), but it does reduce the number of rows in the output stream while only blowing up the space usage by $O(1/\epsilon^2)$ (due to requiring the storage of an ϵ-quality approximation rather than only $O(1)$).

The BSS method maintains two matrices, $\mathbf{B}_i^U$ and $\mathbf{B}_i^L$, acting as upper and lower "barriers". The current spectral approximation will always fall between them:

$$\mathbf{B}_i^L \prec \tilde{\mathbf{A}}_i^T \tilde{\mathbf{A}}_i^T \prec \mathbf{B}_i^U .$$

This guarantee, at the end of the algorithm, will ensure that $\tilde{\mathbf{A}}$ is a valid approximation.

Below, we give the actual BSS algorithm and its performance guarantees.

▶ **Theorem 10.**

1. *The online BSS algorithm always outputs $\tilde{A}$ such that*

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \prec \tilde{\mathbf{A}}^T \tilde{\mathbf{A}}^T \prec (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I} .$$

2. *The probability that a row $\mathbf{a}_i$ is included in $\tilde{\mathbf{A}}$ is at most $\frac{8}{\epsilon^2} l_i$, where l_i is the online $\frac{2\delta}{\epsilon}$-ridge leverage score of $\mathbf{a}_i$. That is $l_i = \min(\mathbf{a}_i^T \left(\mathbf{A}_i^T\mathbf{A}_i + \frac{2\delta}{\epsilon} I\right)^{-1} \mathbf{a}_i, 1)$. The expected number of rows in $\tilde{\mathbf{A}}$ is thus at most $\frac{8}{\epsilon^2}\sum_{i=1}^n l_i = \mathcal{O}(d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$.*

Proof of Theorem 10 part 1. We first note the basic invariant that $\mathbf{X}_i^U$ and $\mathbf{X}_i^L$ always remain positive definite–or equivalently,

$$\mathbf{B}_i^L \prec \tilde{\mathbf{A}}_i^T \tilde{\mathbf{A}}_i^T \prec \mathbf{B}_i^U .$$

We may prove this by induction on i. The base case follows from the initialization of $\tilde{\mathbf{A}}_0$, $\mathbf{B}_0^U$ and $\mathbf{B}_0^L$. For each successive step, we consider two possibilities.

```
$\tilde{\mathbf{A}} = $ ONLINE-BSS$(\mathbf{A}, \epsilon, \delta)$, where $\mathbf{A}$ is an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$,
$\epsilon \in (0,1)$, $\delta > 0$.
1. Set $c_U = \frac{2}{\epsilon} + 1$ and $c_L = \frac{2}{\epsilon} - 1$.
2. Let $\tilde{\mathbf{A}}_0$ be a $0 \times d$ matrix, $\mathbf{B}_0^U = \delta\mathbf{I}$, $\mathbf{B}_0^L = -\delta\mathbf{I}$.
3. For $i = 1, \ldots, n$:
   a. Let $\mathbf{X}_{i-1}^U = (\mathbf{B}_{i-1}^U - \tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1})$, $\mathbf{X}_{i-1}^L = (\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} - \mathbf{B}_{i-1}^L)$.
   b. Let $p_i := \min(c_U\mathbf{a}_i^T(\mathbf{X}_{i-1}^U)^{-1}\mathbf{a}_i + c_L\mathbf{a}_i^T(\mathbf{X}_{i-1}^L)^{-1}\mathbf{a}_i, 1)$.
   c. Set $\tilde{\mathbf{A}}_i := \begin{cases} \begin{bmatrix} \tilde{\mathbf{A}}_{i-1} \\ \mathbf{a}_i/\sqrt{p_i} \end{bmatrix} & \text{with probability } p_i, \\ \tilde{\mathbf{A}}_{i-1} & \text{otherwise.} \end{cases}$
   d. Set $\mathbf{B}_i^U = \mathbf{B}_{i-1}^U + (1+\epsilon)\mathbf{a}_i\mathbf{a}_i^T$, $\mathbf{B}_i^L = \mathbf{B}_{i-1}^L + (1-\epsilon)\mathbf{a}_i\mathbf{a}_i^T$.
4. Return $\tilde{\mathbf{A}} := \tilde{\mathbf{A}}_n$.
```

Figure 3 The Online BSS Algorithm.

The first is that $p_i = 1$. In that case, $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}$ always increases by exactly $\mathbf{a}_i\mathbf{a}_i^T$, $\mathbf{B}^U$ by $(1+\epsilon)\mathbf{a}_i\mathbf{a}_i^T$ and $\mathbf{B}^L$ by $(1-\epsilon)\mathbf{a}_i\mathbf{a}_i^T$. Thus $\mathbf{X}^U$ and $\mathbf{X}^L$ increase by exactly $\epsilon\mathbf{a}_i\mathbf{a}_i^T$, which is positive semidefinite, and so remain positive definite.

In the other case, $p_i < 1$. Now, $\mathbf{X}^U$ decreases by at most the increase in $\tilde{\mathbf{A}}_i^T\tilde{\mathbf{A}}_i^T$, or

$$\mathbf{M}_i = \frac{\mathbf{a}_i\mathbf{a}_i^T}{p}.$$

Since $c_U > 1$, $p > \mathbf{a}_i^T(\mathbf{X}_{i-1}^U)^{-1}\mathbf{a}_i$, so $\mathbf{a}_i\mathbf{a}_i^T \prec p\mathbf{X}_{i-1}^U$ and $\mathbf{M}_i \prec \mathbf{X}_{i-1}^U$. Subtracting this then must leave $\mathbf{X}^U$ positive definite. Similarly, $\mathbf{X}^L$ decreases by at most the increase in $\mathbf{B}^L$, which is $(1-\epsilon)\mathbf{a}_i\mathbf{a}_i^T \prec \mathbf{a}_i\mathbf{a}_i^T$. Since $c_L > 1$ and $p < 1$, $\mathbf{a}_i^T(\mathbf{X}_{i-1}^L)^{-1}\mathbf{a}_i < 1$, and $\mathbf{a}_i\mathbf{a}_i^T \prec \mathbf{X}_{i-1}^L$. Subtracting this similarly leaves $\mathbf{X}^L$ positive definite. Finally, we note that

$$\mathbf{B}_n^U = (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I}$$
$$\mathbf{B}_n^L = (1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I}.$$

This gives the desired result. ◀

To prove part 2, we will use quantities of the form $\mathbf{v}^T\mathbf{X}^{-1}\mathbf{v}$. We will need a lemma describing how this behaves under a random rank-1 update:

▶ **Lemma 11.** *Given a positive definite matrix $\mathbf{X}$, two vectors $\mathbf{u}$ and $\mathbf{v}$, two multipliers a and b and a probability p, define the random variable $\mathbf{X}'$ to be $X - a\mathbf{u}\mathbf{u}^T$ with probability p and $X - b\mathbf{u}\mathbf{u}^T$ otherwise. Then if $\mathbf{u}^T\mathbf{X}^{-1}\mathbf{u} = 1$,*

$$\mathbf{E}\left[\mathbf{v}^T\mathbf{X}'^{-1}\mathbf{v} - \mathbf{v}^T\mathbf{X}^{-1}\mathbf{v}\right] = (\mathbf{v}^\mathbf{T}\mathbf{X}^{-1}\mathbf{u})^2\frac{pa + (1-p)b - ab}{(1-a)(1-b)}\right].$$

Proof. We apply the Sherman-Morrison formula to each of the two possibilities (subtracting $a\mathbf{u}\mathbf{u}^T$ and $b\mathbf{u}\mathbf{u}^T$ respectively). These give $\mathbf{X}'$ values of respectively

$$\mathbf{X}^{-1} + a\frac{\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}}{1 - a\mathbf{u}^T\mathbf{X}^{-1}\mathbf{u}} = \mathbf{X}^{-1} + \frac{a}{1-a}\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}$$

and

$$\mathbf{X}^{-1} + b\frac{\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}}{1 - b\mathbf{u}^T\mathbf{X}^{-1}\mathbf{u}} = \mathbf{X}^{-1} + \frac{b}{1-b}\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}.$$

The values of $\mathbf{v}^T\mathbf{X}'^{-1}\mathbf{v} - \mathbf{v}^T\mathbf{X}^{-1}\mathbf{v}$ are then respectively

$$\frac{a}{1-a}\mathbf{v}^T\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}\mathbf{v} = (\mathbf{v}^\mathbf{T}\mathbf{X}^{-1}\mathbf{u})^2\frac{a}{1-a}$$

and

$$\frac{b}{1-b}\mathbf{v}^T\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}\mathbf{v} = (\mathbf{v}^\mathbf{T}\mathbf{X}^{-1}\mathbf{u})^2\frac{b}{1-b}.$$

Combining these gives the stated result. ◀

Proof of Theorem 10 part 2. First, we introduce some new matrices to help in the analysis:

$$\begin{aligned}\mathbf{C}^U_{i,j} &= \delta\mathbf{I} + \frac{\epsilon}{2}\mathbf{A}_i^T\mathbf{A}_i + \left(1+\frac{\epsilon}{2}\right)\mathbf{A}_j^T\mathbf{A}_j\\ \mathbf{C}^L_{i,j} &= -\delta\mathbf{I} - \frac{\epsilon}{2}\mathbf{A}_i^T\mathbf{A}_i + \left(1-\frac{\epsilon}{2}\right)\mathbf{A}_j^T\mathbf{A}_j.\end{aligned}$$

Note that $\mathbf{C}^U_{i,i} = \mathbf{B}^U_i$, $\mathbf{C}^L_{i,i} = \mathbf{B}^L_i$, and for $j \le i$, $\mathbf{C}^U_{i,j} \succeq \mathbf{B}^U_j$ and $\mathbf{C}^L_{i,j} \preceq \mathbf{B}^L_j$. We can then define:

$$\begin{aligned}\mathbf{Y}^U_{i,j} &= \mathbf{C}^U_{i,j} - \tilde{\mathbf{A}}_j^T\tilde{\mathbf{A}}_j\\ \mathbf{Y}^L_{i,j} &= \tilde{\mathbf{A}}_j^T\tilde{\mathbf{A}}_j - \mathbf{C}^L_{i,j}.\end{aligned}$$

We then have, similarly, $\mathbf{Y}^U_{i,i} = \mathbf{X}^U_i$, $\mathbf{Y}^L_{i,i} = \mathbf{X}^L_i$, and for $j \le i$, $\mathbf{Y}^U_{i,j} \succeq \mathbf{X}^U_j$ and $\mathbf{Y}^L_{i,j} \succeq \mathbf{X}^L_j$.

We will assume that $l_i < 1$, since otherwise the claim is immediate (as probabilities cannot exceed 1). Now, note that

$$\begin{aligned}\mathbf{a}_i^T(\mathbf{Y}^U_{i,0})^{-1}\mathbf{a}_i &= \mathbf{a}_i^T(\mathbf{Y}^L_{i,0})^{-1}\mathbf{a}_i\\ &= \mathbf{a}_i^T\left(\frac{\epsilon}{2}\mathbf{A}_i^T\mathbf{A}_i + \delta I\right)^{-1}\mathbf{a}_i\\ &= \frac{2}{\epsilon}\left(\mathbf{A}_i^T\mathbf{A}_i + \frac{2\delta}{\epsilon}I\right)^{-1}\mathbf{a}_i\\ &= \frac{2}{\epsilon}l_i.\end{aligned}$$

Next, we will aim to show that for $j < i-1$,

$$\begin{aligned}&\mathbf{E}\left[\mathbf{a}_i^T\mathbf{Y}^U_{i-1,j+1}\mathbf{a}_i\right] \le \mathbf{E}\left[\mathbf{a}_i^T\mathbf{Y}^U_{i-1,j}\mathbf{a}_i\right]\\ &\mathbf{E}\left[\mathbf{a}_i^T\mathbf{Y}^L_{i-1,j+1}\mathbf{a}_i\right] \le \mathbf{E}\left[\mathbf{a}_i^T\mathbf{Y}^L_{i-1,j}\mathbf{a}_i\right]\end{aligned}$$

In particular, we will simply show that conditioned on any choices for the first j rows, the expected value of $\mathbf{a}_i^T\mathbf{Y}^U_{i-1,j+1}\mathbf{a}_i$ is no larger than $\mathbf{a}_i^T\mathbf{Y}^U_{i-1,j}\mathbf{a}_i$, and analogously for $\mathbf{Y}^L$.

Similar to the proof of part 1, we separately consider the case where $p_{j+1} = 1$. In that case, the positive semidefinite matrix $\frac{\epsilon}{2}\mathbf{a}_j\mathbf{a}_j^T$ is simply added to $\mathbf{Y}^U$ and $\mathbf{Y}^L$. Adding this can only decrease the values of $\mathbf{a}_i^T\mathbf{Y}^U\mathbf{a}_i$ and $\mathbf{a}_i^T\mathbf{Y}^L\mathbf{a}_i$.

The $p_{j+1} < 1$ case is more tricky. Here, we define the vector $\mathbf{w}_{j+1} = \frac{\mathbf{a}_{j+1}}{\sqrt{p_{j+1}}}$. Importantly

$$\begin{aligned}p_{j+1} &\ge c_U\mathbf{a}_{j+1}^T(\mathbf{X}^U_j)^{-1}\mathbf{a}_{j+1} \ge c_U\mathbf{a}_{j+1}^T(\mathbf{Y}^U_{i-1,j})^{-1}\mathbf{a}_{j+1}\\ p_{j+1} &\ge c_L\mathbf{a}_{j+1}^T(\mathbf{X}^L_j)^{-1}\mathbf{a}_{j+1} \ge c_L\mathbf{a}_{j+1}^T(\mathbf{Y}^L_{i-1,j})^{-1}\mathbf{a}_{j+1}.\end{aligned}$$

This means that

$$\begin{aligned}\mathbf{w}_{j+1}^T(\mathbf{Y}_{i-1,j}^U)^{-1}\mathbf{w}_{j+1}^T &\leq \frac{1}{c_U}\\ \mathbf{w}_{j+1}^T(\mathbf{Y}_{i-1,j}^L)^{-1}\mathbf{w}_{j+1}^T &\leq \frac{1}{c_L}.\end{aligned}$$

Now, we additionally define

$$\begin{aligned}s_{j+1}^U &= \mathbf{w}_{j+1}^T(\mathbf{Y}_{i-1,j}^U)^{-1}\mathbf{w}_{j+1}^T\\ s_{j+1}^L &= \mathbf{w}_{j+1}^T(\mathbf{Y}_{i-1,j}^L)^{-1}\mathbf{w}_{j+1}^T\\ \mathbf{u}_{j+1}^U &= \frac{\mathbf{w}_{j+1}}{\sqrt{s_{j+1}^U}}\\ \mathbf{u}_{j+1}^L &= \frac{\mathbf{w}_{j+1}}{\sqrt{s_{j+1}^L}}.\end{aligned}$$

We then deploy Lemma 11 to compute the expectations. For the contribution from the upper barrier, we use $\mathbf{X} = \mathbf{Y}_{i-1,j}^U$, $\mathbf{u} = \mathbf{u}_{j+1}^U$, $\mathbf{v} = \mathbf{a}_i^T$, $a = -s_{j+1}^U(1 - p_{j+1}(1 + \epsilon/2))$, $b = s_{j+1}^U p_{j+1}(1 + \epsilon/2)$, $p = p_{j+1}$. For the lower barrier, we use $\mathbf{X} = \mathbf{Y}_{i-1,j}^L$, $\mathbf{u} = \mathbf{u}_{j+1}^L$, $\mathbf{v} = \mathbf{a}_i^T$, $a = s_{j+1}^L(1 - p_{j+1}(1 - \epsilon/2))$, $b = -s_{j+1}^L p_{j+1}(1 - \epsilon/2)$, $p = p_{j+1}$. In both cases we can see that the numerator of the expected change is nonpositive. Finally, this implies that the probability that row i is sampled is

$$\begin{aligned}\mathbf{E}\left[p_i\right] &= c_U\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{X}_{i-1}^U)^{-1}\mathbf{a}_i\right] + c_L\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{X}_{i-1}^L)^{-1}\mathbf{a}_i\right]\\ &= c_U\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{Y}_{i-1,i-1}^U)^{-1}\mathbf{a}_i\right] + c_L\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{Y}_{i-1,i-1}^L)^{-1}\mathbf{a}_i\right]\\ &\leq c_U\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{Y}_{i-1,0}^U)^{-1}\mathbf{a}_i\right] + c_L\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{Y}_{i-1,0}^L)^{-1}\mathbf{a}_i\right]\\ &= \frac{2}{\epsilon}(c_U + c_L)l_i\\ &= \frac{8}{\epsilon^2}l_i\end{aligned}$$

as desired. ◀

5 Matching Lower Bound

Here we show that the row count obtained by Theorem 10 is in fact optimal. While it is possible to obtain a spectral approximation with $\mathcal{O}(d/\epsilon^2)$ rows in the offline setting, online sampling always incurs a loss of $\Omega\left(\log(\epsilon\|\mathbf{A}\|_2^2/\delta)\right)$ and must sample $\Omega\left(\frac{d\log(\epsilon\|\mathbf{A}\|_2^2/\delta)}{\epsilon^2}\right)$ rows.

▶ **Theorem 12.** *Assume that $\epsilon\|\mathbf{A}\|_2^2 \geq c_1\delta$ and $\epsilon \geq c_2/\sqrt{d}$, for fixed constants c_1 and c_2. Then any algorithm that selects rows in an online manner and outputs a spectral approximation to $\mathbf{A}^T\mathbf{A}$ with $(1+\epsilon)$ multiplicative error and δ additive error with probability at least $1/2$ must sample $\Omega\left(\frac{d\log(\epsilon\|\mathbf{A}\|_2^2/\delta)}{\epsilon^2}\right)$ rows of $\mathbf{A}$ in expectation.*

Note that the lower bounds we assume on $\epsilon\|\mathbf{A}\|_2^2$ and ϵ are very minor. They just ensure that $\log(\epsilon\|\mathbf{A}\|_2^2/\delta) \geq 1$ and that ϵ is not so small that we can essentially sample all rows.

Proof. We apply Yao's minimax principle, constructing, for any large enough M, a distribution on inputs $\mathbf{A}$ with $\|\mathbf{A}\|_2^2 \leq M$ for which any deterministic online row selection algorithm that succeeds with probability at least $1/2$ must output $\Omega\left(\frac{d\log(\epsilon M/\delta)}{\epsilon^2}\right)$ rows in expectation. The best randomized algorithm that works with probability $1/2$ on any input matrix with $\|\mathbf{A}\|_2^2 \leq M$ therefore must select at least $\Omega\left(\frac{d\log(\epsilon M/\delta)}{\epsilon^2}\right)$ rows in expectation on the worst case input, giving us the theorem.

Our distribution is as follows. We select an integer N uniformly at random from $[1, \log(M\epsilon/\delta)]$. We then stream in the vertex edge incidence matrices of N complete graphs on d vertices. We double the weight of each successive graph. Intuitively, spectrally approximating a complete graph requires selecting $\Omega(d/\epsilon^2)$ edges [2] (as long as $\epsilon \geq c_2/\sqrt{d}$ for some fixed constant c_2). Each time we stream in a new graph with double the weight, we force the algorithm to add $\Omega(d/\epsilon^2)$ more edges to its output, eventually forcing it to output $\Omega(d/\epsilon^2 \cdot N)$ edges – $\Omega(d\log(M\epsilon/\delta)/\epsilon^2)$ in expectation.

Specifically, let $\mathbf{K}_d$ be the $\binom{d}{2} \times d$ vertex edge incidence matrix of the complete graph on d vertices. $\mathbf{K}_d^T\mathbf{K}_d$ is the Laplacian matrix of the complete graph on d vertices. We weight the first graph so that its Laplacian has all its nonzero eigenvalues equal to δ/ϵ. (That is, each edge has weight $\frac{\delta}{d\epsilon}$). In this way, even if we select $N = \lfloor\log(M\epsilon/\delta)\rfloor$ we will have overall $\|\mathbf{A}\|_2^2 \leq \delta/\epsilon + 2\delta/\epsilon + ...2^{\lfloor\log(M\epsilon/\delta)\rfloor-1}\delta/\epsilon \leq M$.

Even if $N = 1$, all nonzero eigenvalues of $\mathbf{A}^T\mathbf{A}$ are at least δ/ϵ, so achieving $(1+\epsilon)$ multiplicative error and $\delta\mathbf{I}$ additive error is equivalent to achieving $(1+2\epsilon)$ multiplicative error. $\mathbf{A}^T\mathbf{A}$ is a graph Laplacian so has a null space. However, as all rows are orthogonal to the null space, achieving additive error $\delta\mathbf{I}$ is equivalent to achieving additive error $\delta\mathbf{I}_r$ where $\mathbf{I}_r$ is the identity projected to the span of $\mathbf{A}^T\mathbf{A}$. $\delta\mathbf{I}_r \preceq \epsilon\mathbf{A}^T\mathbf{A}$ which is why we must achieve $(1+2\epsilon)$ multiplicative error.

In order for a deterministic algorithm to be correct with probability $1/2$ on our distribution, it must be correct for at least $1/2$ of our $\lfloor\log(M\epsilon/\delta)\rfloor$ possible choices of N.

Let i be the lowest choice of N for which the algorithm is correct. By the lower bound of [2], the algorithm must output $\Omega(d/\epsilon^2)$ rows of $\mathbf{A}_i$ to achieve a $(1+2\epsilon)$ multiplicative factor spectral approximation. Here $\mathbf{A}_i$ is the input consisting of the vertex edge incidence matrices of i increasingly weighted complete graphs. Call the output on this input $\tilde{\mathbf{A}}_i$. Now let j be the second lowest choice of N on which the algorithm is correct. Since the algorithm was correct on $\mathbf{A}_i$ to within a multiplicative $(1+2\epsilon)$, to be correct on $\mathbf{A}_j$, it must output a set of edges $\tilde{\mathbf{A}}_j$ such that

$$(\mathbf{A}_j^T\mathbf{A}_j - \mathbf{A}_i^T\mathbf{A}_i) - 4\epsilon\mathbf{A}_j^T\mathbf{A}_j \preceq \tilde{\mathbf{A}}_j^T\tilde{\mathbf{A}}_j - \tilde{\mathbf{A}}_i^T\tilde{\mathbf{A}}_i \preceq (\mathbf{A}_j^T\mathbf{A}_j - \mathbf{A}_i^T\mathbf{A}_i) + 4\epsilon\mathbf{A}_j^T\mathbf{A}_j.$$

Since we double each successive copy of the complete graph, $\mathbf{A}_j^T\mathbf{A}_j \preceq 2(\mathbf{A}_j^T\mathbf{A}_j - \mathbf{A}_i^T\mathbf{A}_i)$. So, $\tilde{\mathbf{A}}_j^T\tilde{\mathbf{A}}_j - \tilde{\mathbf{A}}_i^T\tilde{\mathbf{A}}_i$ must be a $1+8\epsilon$ spectral approximation to the true difference $\mathbf{A}_j^T\mathbf{A}_j - \mathbf{A}_i^T\mathbf{A}_i$. Noting that this difference is itself just a weighting of the complete graph, by the lower bound in [2] the algorithm must select $\Omega(d/\epsilon^2)$ additional edges between the i^{th} and j^{th} input graphs. Iterating this argument over all $\lfloor\log(M\epsilon/\delta)\rfloor/2$ inputs on which the algorithm must be correct, it must select a total of $\Omega(d\log(M\epsilon/\delta)/\epsilon^2)$ edges in expectation over all inputs. ◀

6 Future Work

An obvious open question arising from our work is if one can prove that the algorithm of [12] works despite dependencies arising due to the row pruning step. By operating in the online setting, our algorithm avoids row pruning, and hence is able to skirt these dependencies, as the probability that a row is sampled only depends on earlier rows in the stream. However, because the streaming setting offers the potential for sampling fewer rows than in the online case, obtaining a rigorous proof of [12] would be very interesting.

While our work focuses on spectral approximation, variants on (ridge) leverage score sampling and the BSS algorithm are also used to solve low-rank approximation problems, including column subset selection [5, 9] and projection-cost-preserving sketching [7, 9]. Compared with spectral approximation, there is less work on streaming sampling for low-rank approximation, and understanding how online algorithms may be used in this setting would an interesting extension of our work.

Acknowledgements. The authors would like to thank Kenneth Clarkson, Jonathan Kelner, Gary Miller, Christopher Musco and Richard Peng for helpful discussions and comments.

Cameron Musco and Jakub Pachocki both acknowledge the Gene Golub SIAM Summer School program on Randomization in Numerical Linear Algebra, where work on this project was initiated.

References

1 Ahmed Alaoui and Michael W Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 775–783, 2015.

2 Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.

3 Antoine Bordes and Léon Bottou. The huller: a simple and efficient online SVM. In *Machine Learning: ECML 2005*, pages 505–512. Springer, 2005.

4 Christos Boutsidis, Dan Garber, Zohar Karnin, and Edo Liberty. Online principal components analysis. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 887–901, 2015.

5 Christos Boutsidis and David P Woodruff. Optimal CUR matrix decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 353–362, 2014.

6 Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2013.

7 Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–172, 2015.

8 Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 6th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 181–190, 2015.

9 Michael B Cohen, Cameron Musco, and Christopher Musco. Ridge leverage scores for low-rank approximation. *arXiv:1511.07263*, 2015.

10 Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

11 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 561–570, 2014.

12 Jonathan A Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *Theory of Computing Systems*, 53(2):243–262, 2013.

13 Ioannis Koutis, Gary L Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 235–244, 2010.

14 Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 250–269, 2015.

15 Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 127–136, 2013.

16 Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online k-means clustering. In *Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 81–89, 2016.

17 Michael W. Mahoney and Xiangrui Meng. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 91–100, 2013.

18 Jelani Nelson and Huy L. Nguyen. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 117–126, 2013.

19 Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

20 Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2004.

21 Daniel A Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014.

22 Joel Tropp. Freedman's inequality for matrix martingales. *Electronic Communications in Probability*, 16:262–270, 2011.

Oblivious Rounding and the Integrality Gap*

Uriel Feige†1, Michal Feldman‡2, and Inbal Talgam-Cohen§3

1 **Department of Computer Science, Weizmann Institute of Science, Israel**
`uriel.feige@weizmann.ac.il`

2 **School of Computer Science, Tel-Aviv University, and Microsoft Research, Israel**
`michal.feldman@cs.tau.ac.il`

3 **School of Computer Science, Hebrew and Tel-Aviv Universities, Israel**
`inbal.talgamcohen@mail.huji.ac.il`

Abstract

The following paradigm is often used for handling NP-hard combinatorial optimization problems. One first formulates the problem as an integer program, then one relaxes it to a linear program (LP, or more generally, a convex program), then one solves the LP relaxation in polynomial time, and finally one rounds the optimal LP solution, obtaining a feasible solution to the original problem. Many of the commonly used rounding schemes (such as randomized rounding, threshold rounding and others) are *oblivious* in the sense that the rounding is performed based on the LP solution alone, disregarding the objective function. The goal of our work is to better understand in which cases oblivious rounding suffices in order to obtain approximation ratios that match the integrality gap of the underlying LP. Our study is information theoretic – the rounding is restricted to be oblivious but not restricted to run in polynomial time. In this information theoretic setting we characterize the approximation ratio achievable by oblivious rounding. It turns out to equal the integrality gap of the underlying LP on a problem that is the *closure* of the original combinatorial optimization problem. We apply our findings to the study of the approximation ratios obtainable by oblivious rounding for the maximum welfare problem, showing that when valuation functions are submodular oblivious rounding can match the integrality gap of the configuration LP (though we do not know what this integrality gap is), but when valuation functions are gross substitutes oblivious rounding cannot match the integrality gap (which is 1).

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Welfare-maximization

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.8

1 Introduction

Rounding and Obliviousness

Consider a combinatorial maximization problem π, represented by a pair (V, X). The set V contains all possible problem *instances*, where an instance is the linear objective function to

* This work was partly done at Microsoft Research, Herzliya, Israel.
† The work of U. Feige was supported in part by the Israel Science Foundation (grant No. 621/12) and by the I-CORE Program of the Planning and Budgeting Committee and the Israel Science Foundation (grant no. 4/11).
‡ The work of M. Feldman was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement number 337122.
§ The work of I. Talgam-Cohen was supported in part by the I-CORE Program of the Planning and Budgeting Committee and the Israel Science Foundation (grant no. 4/11), and by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement number 337122.

be maximized, represented as a vector in $\mathbb{R}^d_{\geq 0}$. The set X contains all feasible *solutions* to the problem, also represented as vectors in $\mathbb{R}^d_{\geq 0}$. The goal is, given an instance $v \in V$, to return a feasible solution $x \in X$ that maximizes the objective $v \cdot x$ among all feasible solutions. If the combinatorial problem is hard, the goal is to approximate rather than optimize the objective. As a concrete example, consider the problem of finding a max-cut in a complete weighted graph. In this case, V is the set of all possible edge weights, and X is the set of all valid cuts where each cut is represented by the set of edges in the cut. The objective value $v \cdot x$ that a cut x obtains for a weighted graph v is the total weight of edges in the cut.

A paradigmatic approach to solving combinatorial optimization problems is that of *relaxation and rounding*: The problem π is relaxed to a new problem $\pi' = (V, Y)$ where Y is such that $X \subset Y$, i.e., the new feasible solution set is a relaxation of the original one. Typically the new feasible domain is fractional while the original one is integral. To solve a given instance v, first a relaxed solution $y \in Y$ to the new problem π' is computed, and then it is (randomly) rounded to get a solution $x \in X$ to the original problem π. The algorithm designer aims to design a rounding process that does not lose too much in the objective value, i.e., for which the inner product $v \cdot x$ is not far in a multiplicative sense (and in expectation) from $v \cdot y$. If she succeeds we say that the rounding scheme guarantees a good *approximation ratio*. A rounding scheme is *oblivious* if y is rounded to x without knowledge of the objective function v.[1] In other words, v is used only to obtain a relaxed solution (e.g., to formulate and solve a linear program), and not to round it back to a feasible solution (e.g., a solution to a corresponding integer program).

Many rounding schemes in the optimization literature are oblivious, and many are not oblivious (see Section 5). This raises the following natural question: Is there a reason why for some problems oblivious rounding works well (achieves good approximation ratios to the optimal objective), while for others it fails miserably? For an algorithm designer it may be very useful to be able to predict in advance whether the relaxation she has formulated for the problem admits an oblivious rounding scheme with a good approximation ratio, or whether any good scheme will need to utilize the objective function to guide its rounding process. *The purpose of this paper is to initiate a systematic study of the power of oblivious rounding relative to its non-oblivious counterpart.* We study this question from an *information* perspective, imposing no polynomial time constraint on the rounding schemes. We remark that even non-polynomial time rounding schemes are of interest, for example, as a way of bounding the integrality gap of the underlying relaxation.

Advantages of Oblivious Rounding

There is also reason to try and aim specifically for a relaxation that admits good oblivious rounding, and/or to be able to prove the impossibility of getting a good approximation via oblivious rounding. The advantages of rounding that is oblivious are demonstrated nicely in the context of *welfare maximization in combinatorial auctions*, which will be the main domain in which we demonstrate the results of our study of oblivious rounding (see Section 4 for more details on welfare maximization). In this context, indivisible items are to be allocated among buyers, each with her own valuation function mapping bundles of items to values. The valuations are very large objects (exponential in the number of items), and there is extensive literature related to their communication complexity (see, e.g., [20]). Oblivious

[1] Our notion of oblivious rounding is not to be confused with the rounding technique of [27], which avoids solving a linear program – see the discussion of related work below.

rounding limits the algorithmic stage in which communication in required, and there is no need for communication after a relaxed solution is found. Also, as we show in Proposition 23, oblivious rounding gives the different buyers the same treatment in terms of the value they are guaranteed to obtain after the rounding, and so has a "built-in" fairness guarantee.

Recently in [7], oblivious rounding was studied in the context of incentive properties of allocation mechanisms. It turns out that when an algorithm is based on the relax-and-round paradigm, and the rounding is oblivious, there are price rules that can be added to the algorithm such that the worst equilibrium behavior (the price of anarchy) is determined by the relaxation and by the approximation ratio of the oblivious rounding. This is quite remarkable, as there is no a priori reason to believe that the consequences of strategic behavior would be determined by algorithmic properties of the rounding, and indeed this is not the case for non-oblivious rounding. Thus, an algorithmic mechanism designer may aim for a design based on obliviousness to get good strategic properties, and so it would be helpful to understand what a design based on oblivious rounding can hope to achieve.

Finally, in [12, 24], the issue of robustness of the welfare guarantees to noise in the objective function is studied. Ideally, an algorithm for approximating welfare will get a good approximation despite small perturbations in the buyers' valuations. In the common case that the welfare maximization problem is relaxed to a linear program which is then solved and rounded, it turns out that solving the LP is quite robust, and so if the rounding is oblivious this ensures the robustness of the entire algorithm.

Our Results

Consider a problem $\pi = (V, X)$ and a relaxed problem $\pi' = (V, Y)$. Our main result is to relate the approximation ratio achievable by oblivious rounding to the well-studied notion of *integrality gap*.

In the context of our work, we define the approximation ratio to be the worst case ratio, over all instances $v \in V$ and all relaxed solutions $y \in Y$, between the (expected) objective value $v \cdot x$ achieved by the (random) rounded solution $x \in X$, and between the objective $v \cdot y$ achieved by the relaxed solution to be rounded y. Note that there is another notion of approximation ratio, which compares $v \cdot x$ achieved by the rounding to $v \cdot x^*$ (rather than $v \cdot y$), where x^* is the optimal feasible solution to instance v. While different in general, in many cases the two notions coincide.

On the other hand, recall that the integrality gap is the worst case ratio, over all instances v and all relaxed solutions y, between the objective value $v \cdot x$ that can be achieved by the *best* feasible solution x, and between $v \cdot y$. As our starting point, we observe that no oblivious rounding can guarantee a better approximation factor than the integrality gap. Thus the question that we ask is: *For which problems does the approximation ratio achievable by oblivious rounding techniques match the integrality gap?* We stress that we do not require oblivious rounding to be polynomial time, but nevertheless the question is of interest due to the information-theoretic obliviousness requirement. This question also comes in another flavor, where one gets an *optimal* solution to the relaxed problem and needs to round it.

Our general results can be summarized informally by the following theorem. The *convex closure* of a problem π is obtained by taking the convex closure of its instance set. This may not actually change the problem, i.e., the instance set may be closed under convex combinations. For example, welfare maximization in a combinatorial auction setting with *submodular* valuations is an example of such a problem (because submodularity is preserved under convex combinations), but with *gross substitutes* (*GS*) valuations it is not (the average of two GS functions need not be GS – see Section 4). The convex closure of the class of GS valuations is the class *cone GS* (*CGS*) defined in [6].

▶ **Theorem** (Informal).

- *For optimization problems closed under convex combinations, the approximation ratio of the best oblivious rounding scheme equals the integrality gap.*
- *More generally, for optimization problems that are not closed under convex combinations, the approximation ratio of the best oblivious rounding scheme equals the integrality gap of the convex closure of the problem.*
- *If the relaxed solution to be rounded obliviously is guaranteed to be optimal, the approximation ratio of the best oblivious rounding scheme is at least the integrality gap of the convex closure of the problem, and may be strictly greater than it in some cases.*

See Section 3 for formal statements of these results.

We apply our general results to the welfare maximization problem for combinatorial auctions. In particular, we use the integrality gap of welfare maximization with *coverage* valuations – the convex closure of unit-demand valuations – to establish a bound on what the best oblivious rounding can achieve for unit-demand valuations.

▶ **Theorem.** *For the welfare maximization problem with unit-demand valuations and for its relaxation based on the* configuration linear program*, no oblivious rounding can get more than a* $5/6$*-approximation ratio for two buyers, and no oblivious rounding can get more than a* 0.782*-approximation ratio for* n *buyers. These bounds immediately extend to gross substitutes, for which the integrality gap is known to be 1.*

Another application of our general results to welfare maximization is the prediction that the above gap, which occurs between the integrality gap and the approximation ratio of the best oblivious rounding for unit-demand valuations, will *not* occur for classes of valuations like submodular valuations, which are closed under convex combinations. See Section 4 for formal statements of these results.

Related Work

In recent years, the connection between various notions of rounding and algorithmic mechanism design has been studied in several works. [15] use the technique of randomized metarounding [3] to derive truthful-in-expectation mechanisms. They require their rounding-based approximation algorithms to satisfy a stronger property than obliviousness (the output expected allocation should be a scaled version of the input for a universal scaling factor). We have already mentioned the work of [7] above, which is directly related to the notion of oblivious rounding that we study (see also Proposition 23 below). Both works and the latter in particular can be seen as strong motivation to systematically study oblivious rounding. [6] requires a different property – convexity of the rounding – in order to derive truthful-in-expectation mechanisms.

In terms of techniques, our work is related to that of [9], which considers a class of *oblivious algorithms* for the max *directed cut* problem. These are algorithms in which each vertex independently decides at random on which side of the cut to place itself, based only on its own in-degree and its own out-degree. One of the results in that work (Theorem 1.8 in the journal version, Theorem 1.5 in the preliminary version) shows equivalence in the worst case approximation ratio of two different ways of using a finite set of oblivious algorithms, one called *mixed* (in which an algorithm is chosen at random), the other called *max* (in which the best algorithm is chosen). The proof of that theorem and the proof of our main theorem are based on similar principles.

[27] introduced a technique for developing approximation algorithms that avoid the bottleneck of first solving a linear program. This technique is also known as "oblivious rounding", but this notion is different than our definition of (objective-)oblivious rounding.

Examples of oblivious rounding techniques that appear in the literature are mentioned in Section 5.

Organization

In Section 2 we present our general framework. In Section 3 we formally state and prove our results for the general framework. Section 4 contains our results for the application of welfare maximization. In Section 5 we list known rounding techniques from the literature and how they fit into the framework.

2 Framework

In this section we present our framework. After several general definitions, in Section 2.1 we define an optimization problem and its relaxation, and recall the well-known notion of integrality gap – a measure of how "relevant" optimization of the relaxation is to optimization of the original problem. In Section 2.2 we introduce oblivious rounding and define the approximation ratio of such rounding schemes, according to how well they round a solution to the relaxed problem into a feasible solution of the original problem.

Let $d \in \mathbb{N}_{>0}$ be a positive integer. For every set $S \subseteq \mathbb{R}^d$ of d-dimensional vectors, let $C(S)$ denote its *convex hull*, i.e., $C(S) = \{\sum_{s \in S} \lambda_s s \mid \forall s \in S : \lambda_s \geq 0 \text{ and } \sum_{s \in S} \lambda_s = 1\}$. A set S is *compact* if it is *closed* (no infinite sequence of vectors converges to a vector outside the set), and *bounded* (there is some finite μ such that the norm of every vector in the set is at most μ). If S is convex and compact, let $\partial(S)$ denote its outer *boundary*, i.e., $\partial(S) = \{s \in S \mid \forall \text{ scalar } \delta \in \mathbb{R}, \delta > 1 : \delta s \notin S\}$.

For sets $S_1, S_2 \subseteq \mathbb{R}^d$, we use the notation $\min_{s_1 \in S_1} \max_{s_2 \in S_2}\{\cdot\}$ when we are optimizing by first choosing $s_1 \in S_1$, and then choosing $s_2 \in S_2$ based on knowledge of s_1; similarly, the notation $\max_{s_2 \in S_2} \min_{s_1 \in S_1}\{\cdot\}$ means that $s_2 \in S_2$ is chosen first and $s_1 \in S_1$ is chosen with prior knowledge of s_2. Here, min and max can be replaces by inf and sup where needed.

2.1 Problems, Relaxations, Closures

We consider optimization problems with linear objectives. We define a problem of dimension d as a collection of d-dimensional instances coupled with a feasible solution set. This means that in our formulation, problem instances of a certain dimension all share the same set of feasible solutions.

For concreteness our framework is developed for *maximization* problems (the results can be adapted also to minimization).

▶ **Definition 1.** A *problem* π of dimension d is a pair (V, X), where $V, X \subseteq \mathbb{R}^d_{\geq 0}$ are nonempty sets of d-dimensional vectors with non-negative entries. V contains the problem *instances* (also called *value functions* or *objectives*), and X is the set of feasible *solutions*. Given an instance $v \in V$, the *value* of solution $x \in X$ is the inner product $v \cdot x$, and x is *optimal* if it has maximum value among all feasible solutions.[2]

[2] The non-negativity in this definition of vectors in V, X can be replaced by a weaker condition of $v \cdot x \geq 0$ for every $v \in V, x \in X$, and our results will still hold.

For concreteness, recall the max-cut example mentioned in Section 1: The instances are modeled as weighted complete graphs over n nodes, all of which share the same set of possible cuts. An instance is thus simply a vector of $n(n-1)/2$ non-negative edge weights, and a feasible solution is a $\{0,1\}$-vector indicating the edges that participate in a cut.

We now define a problem relaxation, which is itself a problem achieved by expanding the original set of feasible solutions:

▶ **Definition 2.** A problem $\pi' = (V, Y)$ is a *relaxation* of problem $\pi = (V, X)$ if $X \subseteq Y$. The solutions in Y are referred to as *relaxed* solutions.

For every relaxed solution $y \in Y$, V_y^+ denotes all instances for which the value of y is strictly positive, and V_y^* denotes all instances for which y is optimal:

$$V_y^+ = \{v \in V \mid v \cdot y > 0\}; \quad V_y^* = \{v \in V \mid v \cdot y \geq v \cdot y' \ \forall y' \in Y\}. \tag{1}$$

Finally, we introduce the closure of a problem, achieved by convexifying the set of instances:

▶ **Definition 3.** The problem $\mathrm{cl}(\pi) = (C(V), X)$ is the *closure* of problem $\pi = (V, X)$.

2.1.1 Assumed Properties of Problems and Relaxations

All problems and relaxations we consider in this paper are assumed to have the natural properties of *compactness* and *positivity* unless stated otherwise, and all relaxations are assumed to be *convex*:

- A problem $\pi = (V, X)$ is *compact* if the feasible solution set X is compact, and there is a compact set $V' \subseteq \mathbb{R}^d_{\geq 0} \setminus \{0^d\}$ such that the instance set V is $\{v = cv' \mid c \in \mathbb{R}_{>0}$ and $v' \in V'\}$. Without loss of generality, the vectors in V' can also be assumed to be normalized (i.e., $\sum_k v'_k = 1$). This is a weaker assumption than assuming V is compact, since it allows unbounded instances as well as instances that approach, but do not reach, 0^d. Many common optimization problems, for example max-cut, are compact: Indeed, the solution set (cuts) is usually closed and bounded; the value functions that make up the instances (edge weights) usually exclude the zero function $v = 0^d$, and so can be normalized as above without loss of generality (without affecting multiplicative approximation factors). Thus V' can be taken to be the set of normalized instances, which is bounded and closed.[3]
- A problem $\pi = (V, X)$ is *positive* if for every $v \in V$ there is some $x \in X$ such that $v \cdot x > 0$ (in particular, V is not allowed to include 0^d), and for every $x \in X \setminus \{0^d\}$ there is some $v \in V$ such that $v \cdot x > 0$. In the max-cut example, the first positivity condition holds because $v \neq 0^d$ and so at least one edge must have nonzero weight. For the second positivity condition, a natural sufficient condition is that the graph has a spanning tree such that for every edge in the tree, there is an edge-weight function in V that assigns positive weight to that edge. For every cut x there is at least one edge of the spanning tree in the cut, and therefore at least one instance v such that $v \cdot x > 0$. Notice that by the positivity assumption applied to a relaxation π', V_y^+ is nonempty for every $y \in Y \setminus \{0^d\}$, ensuring that our definitions (such as Definition 4 below) are well-defined.
- A relaxation $\pi' = (V, Y)$ to problem $\pi = (V, X)$ is *convex* if the set Y of relaxed solutions is convex. For example, relaxations that result from formulating the problem as an integer

[3] There is also a version of our results that holds when V' is not closed, in which sup and inf replace max and min in the appropriate places.

program and relaxing it to a linear program are convex. If π' is convex then in particular Y includes the convex hull $C(X)$.

Observe that if a problem is compact and positive, then its closure is also compact and positive.

2.1.2 Integrality Gap

Given a problem $\pi = (V, X)$ and a relaxation $\pi' = (V, Y)$, an important measure of the quality of the relaxation is the integrality gap – the worst case (smallest) ratio, over all possible instances in V, between the value achievable for the instance by a feasible solution in X, and the value achievable for it by a relaxed solution in Y. Formally:

▶ **Definition 4.** Let $\pi = (V, X)$ and $\pi' = (V, Y)$ be a problem and its relaxation. For every relaxed solution $y \in Y \setminus \{0^d\}$ and instance $v \in V_y^+$, the *integrality gap at v, y* is

$$\rho_{\pi,\pi'}(v, y) = \max_{x \in X} \frac{v \cdot x}{v \cdot y}.$$

The integrality gap at solution y is then obtained by taking the worst case instance v, i.e., $\rho_{\pi,\pi'}(y) = \inf_{v \in V_y^+} \rho_{\pi,\pi'}(v, y)$. Similarly, the integrality gap at instance v is $\rho_{\pi,\pi'}(v) = \inf_{y : v \in V_y^+} \rho_{\pi,\pi'}(v, y)$. The (overall) integrality gap is $\rho_{\pi,\pi'} = \inf_{y \in Y \setminus \{0^d\}} \rho_{\pi,\pi'}(y)$.

We make several basic observations regarding the integrality gap. Short proofs appear for completeness in Appendix A.

▶ **Observation 5.** *The integrality gap $\rho_{\pi,\pi'}$ is ≤ 1.*

Informally, the closer $\rho_{\pi,\pi'}$ is to 1, the better the relaxation.

Taking the closure of a problem expands the instance set and so makes it "harder" to get a good relaxation:

▶ **Observation 6.** *For every π and relaxation π', $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')} \leq \rho_{\pi,\pi'}$.*

The next observation shows that to find the integrality gap, we may restrict attention to relaxed solutions that lie on the boundary. Recall that Y is compact, then:

▶ **Observation 7.** *The overall integrality gap is not affected by the integrality gaps at relaxed solutions that lie strictly within the boundary: $\rho_{\pi,\pi'} = \min_{y \in \partial(Y)} \rho_{\pi,\pi'}(y)$.*

2.2 Oblivious Rounding

For the definitions in this section, fix a problem $\pi = (V, X)$ and a relaxation $\pi' = (V, Y)$.

A (randomized) rounding scheme receives an instance $v \in V$ and a relaxed solution $y \in Y$, and returns a distribution over feasible solutions in X. Note that since our objective functions in V are linear, any distribution over feasible solutions in X can be summarized by its average, which lies in the convex hull $C(X)$. This leads to the following definition:

▶ **Definition 8.** A *rounding scheme* is a function $f : V \times Y \to C(X)$.

A rounding scheme is oblivious if it is not allowed to "see" the objective function when rounding a solution of the relaxed problem:

▶ **Definition 9.** An *oblivious* rounding scheme is a function $f : Y \to C(X)$.

▶ Remark. The rounding schemes we consider, whether oblivious or not, need not be computable in polynomial time.

2.2.1 Approximation Ratio of Oblivious Rounding

Our goal is to study the power of oblivious rounding schemes for approximation. For this we shall use the following definition – the approximation ratio of an oblivious rounding scheme is the worst case ratio, over all possible instances in V, between the value achieved for the instance by a rounded solution in X, and the value achievable for it by the corresponding relaxed solution in Y. Formally:

▶ **Definition 10.** Consider an oblivious rounding scheme $f : Y \to C(X)$. For every relaxed solution $y \in Y \setminus \{0^d\}$, the *approximation ratio of f at y* is

$$\alpha_{\pi,\pi'}(y) = \inf_{v \in V_y^+} \frac{v \cdot f(y)}{v \cdot y}.$$

The approximation ratio of f is $\alpha_{\pi,\pi'} = \inf_{y \in Y \setminus \{0^d\}} \alpha_{\pi,\pi'}(y)$.

A larger approximation ratio indicates better approximation by the rounding scheme. A basic observation regarding the approximation ratio is that it is upper-bounded by the integrality gap. A short proof appears in Appendix A for completeness.

▶ **Observation 11.** *For every $y \in Y \setminus \{0^d\}$, the approximation ratio of f at y is at most the integrality gap at y: $\alpha_{\pi,\pi'}(y) \le \rho_{\pi,\pi'}(y)$. Therefore $\alpha_{\pi,\pi'} \le \rho_{\pi,\pi'} \le 1$.*

Observation 11 upper-bounds the approximation ratio, and a natural class of interest is rounding schemes for which this bound is tight:

▶ **Definition 12.** An oblivious rounding scheme $f : Y \to C(X)$ is *tight* if $\alpha_{\pi,\pi'} = \rho_{\pi,\pi'}$, and *individually tight* if $\alpha_{\pi,\pi'}(y) = \rho_{\pi,\pi'}(y)$ for every relaxed solution $y \in Y \setminus \{0^d\}$.

By definition, individual tightness implies tightness.

2.2.2 Approximation Ratio for Optimal Solutions

We are also interested in the approximation guarantees of oblivious rounding schemes *only* for relaxed solutions $y \in Y$ which have the following promised property: they are known to be optimal solutions to some instance of the relaxed problem. Recall from (1) that V_y^* denotes the set of all instances for which y is an optimal solution.

▶ **Observation 13.** *If V_y^* is nonempty then $y \in \partial(Y)$.*

See Appendix A for a proof.

The two definitions in this subsection are analogous to Definitions 10 (approximation ratio) and 12 (tightness) above:

▶ **Definition 14.** Consider an oblivious rounding scheme $f : Y \to C(X)$. For every relaxed solution $y \in Y$ for which $V_y^* \ne \emptyset$, the approximation ratio *for optimal solutions* of f at y is

$$\alpha^*_{\pi,\pi'}(y) = \inf_{v \in V_y^*} \frac{v \cdot f(y)}{v \cdot y}.$$

The approximation ratio for optimal solutions of f is $\alpha^*_{\pi,\pi'} = \inf_{y \in Y : V_y^* \ne \emptyset} \{\alpha^*_{\pi,\pi'}(y)\}$.

By definition, for every $y \in Y$ with nonempty V_y^* it holds that $\alpha_{\pi,\pi'}(y) \le \alpha^*_{\pi,\pi'}(y)$, and so $\alpha_{\pi,\pi'} \le \alpha^*_{\pi,\pi'}$. Note that this inequality may be strict in some cases, and moreover it is not necessarily the case that the upper bound $\rho_{\pi,\pi'}$ on $\alpha_{\pi,\pi'}$ is also an upper bound on $\alpha^*_{\pi,\pi'}$ (see Example 35 below). This motivates the next definition:

▶ **Definition 15.** An oblivious rounding scheme $f : Y \to C(X)$ is tight *for optimal solutions* if $\alpha^*_{\pi,\pi'} \geq \rho_{\pi,\pi'}$, and individually tight *for optimal solutions* if $\alpha^*_{\pi,\pi'}(y) \geq \rho_{\pi,\pi'}(y)$ for every relaxed solution $y \in Y$ with nonempty V^*_y.

By definition, individual tightness for optimal solutions implies tightness for optimal solutions.

3 General Results

In this section we state our results for the general framework; some proofs are deferred to Appendix B.1. Appendix B.2 discusses implications for oblivious rounding of optimal solutions. Additional results that concern the applications of the framework to welfare maximization appear in Section 4.

Recall that the closure of problem $\pi = (V, X)$ is $\mathrm{cl}(\pi) = (C(V), X)$ (Definition 3). Our main general theorem relates the (pointwise) approximation ratio of oblivious rounding to the integrality gap of the problem's closure:

▶ **Theorem 16.** *Given a problem $\pi = (V, X)$ and a relaxation $\pi' = (V, Y)$:*
1. *Upper bound: For every oblivious rounding scheme $f : Y \to C(X)$, at every point $y \in Y \setminus \{0^d\}$, the approximation ratio $\alpha_{\pi,\pi'}(y)$ is at most the integrality gap $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y)$ of the closure of problem π.*
2. *Tightness: There exists an oblivious rounding scheme $f : Y \to C(X)$ such that $\alpha_{\pi,\pi'}(y) = \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y)$ for every $y \in Y \setminus \{0^d\}$.*

Moreover, our proof method yields the following proposition, by which the approximation ratio and integrality gap are achieved by the same instance and (random) feasible solution:

▶ **Proposition 17.** *Given a problem $\pi = (V, X)$, a relaxation $\pi' = (V, Y)$ and a relaxed solution $y \in Y \setminus \{0^d\}$, there exist an instance $v \in C(V)$ and a random feasible solution $x \in C(X)$ of the problem $\mathrm{cl}(\pi)$ such that $\frac{v \cdot x}{v \cdot y} = \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y) = \alpha_{\pi,\pi'}(y)$, where the approximation ratio is that of the best oblivious rounding scheme at y.*

Proof. By Lemma 33. ◀

Two useful corollaries follow immediately from Theorem 16. First, we have already observed that $\alpha_{\pi,\pi'} \leq \rho_{\pi,\pi'}$ (Observation 11) and that $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')} \leq \rho_{\pi,\pi'}$ (Observation 6). It follows from Theorem 16 that for the best oblivious rounding scheme in fact $\alpha_{\pi,\pi'} = \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}$.

▶ **Corollary 18.** *Given a problem $\pi = (V, X)$ and a relaxation $\pi' = (V, Y)$, there exists an oblivious rounding scheme $f : Y \to C(X)$ that achieves an approximation ratio of $\alpha_{\pi,\pi'} = \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}$, and this is the best possible approximation ratio of any oblivious rounding scheme.*

Proof. By Definitions 4 (integrality gap) and 10 (approximation ratio), if for an oblivious rounding scheme f it holds that $\alpha_{\pi,\pi'}(y) = \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y)$ for every $y \in Y \setminus \{0^d\}$, then $\alpha_{\pi,\pi'} = \inf_{y \in Y \setminus \{0^d\}} \alpha_{\pi,\pi'}(y) = \inf_{y \in Y \setminus \{0^d\}} \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')} = \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}$. By Theorem 16 there exists such an oblivious rounding scheme. ◀

▶ **Corollary 19.** *Given a problem $\pi = (V, X)$ whose instances form a convex set (i.e., $\pi = \mathrm{cl}(\pi)$), for every relaxation $\pi' = (V, Y)$, there exists an oblivious rounding scheme $f : Y \to C(X)$ that is individually tight.*

Proof. By Theorem 16 there exists an oblivious rounding scheme f such that $\alpha_{\pi,\pi'}(y) = \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y)$ for every $y \in Y \setminus \{0^d\}$, and by assumption, $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y) = \rho_{\pi,\pi'}(y)$. The proof follows from the definition of individual tightness (Definition 12). ◀

Unlike the statement in Corollary 18, Example 35 shows that the approximation ratio *for optimal solutions* $\alpha^*_{\pi,\pi'}$ may surpass the integrality gap of the closure $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}$. See Appendix B.2 for more details on oblivious rounding of optimal solutions.

3.1 Proof of Theorem 16 via Minimax

Our goal in this section is to prove Theorem 16 via our main lemma (Lemma 33), which is a version of von Neumann's minimax theorem. In the proof we shall use the classic minimax theorem for *non-finite* zero-sum games:

▶ **Theorem 20** ([26]). *For every bipartite zero-sum game in which the players' pure strategy sets X and V are compact and the payoff function $g : \mathcal{V} \times \mathcal{X} \to \mathbb{R}$ is continuous, there exists a unique minimax value μ^* such that*

$$\mu^* = \max_{x \in C(X)} \min_{v \in V} g(v, x) = \min_{v \in C(V)} \max_{x \in X} g(v, x). \tag{2}$$

Moreover, there are equilibrium strategies *$x^* \in C(X)$ and $v^* \in C(V)$ such that x^* maximizes $g(v^*, x)$, v^* minimizes $g(v, x^*)$, and $\mu^* = g(v^*, x^*)$.*

▶ Remark. Throughout this section we shall assume that every problem $\pi = (V, X)$ has a compact instance set V in which instances are normalized (i.e., $\sum_k v_k = 1$). This assumption is without loss of generality, as we assumed in Section 2.1.1 that $V = \{v = cv' \mid c \in \mathbb{R}_{>0} \text{ and } v' \in V'\}$ where V' is compact and normalized. Since an instance $v \in V$ appears exclusively within the expressions $\frac{v \cdot x}{v \cdot y}$ or $\frac{v \cdot f(y)}{v \cdot y}$, the multiplying constant c cancels out and we may as well assume that $V = V'$.

We begin with an intuitive (albeit imprecise) explanation of the connection between the minimax theorem and the approximation ratio of an oblivious rounding scheme. Fix a problem $\pi = (V, X)$, a relaxation $\pi' = (V, Y)$ and a relaxed solution y. We claim that an oblivious rounding scheme f, which maximizes the approximation ratio $\alpha_{\pi,\pi'}(y)$ at y, is equivalent to an optimal mixed strategy in the following zero-sum game (the games used in the actual proof are slightly different): Given y, the maximizing "rounding" player picks a mixed strategy $f(y) \in C(X)$ over feasible solutions in X, and the minimizing "instance" player picks an instance $v \in V$ as his pure strategy best-response to $f(y)$. The expected payoff of the rounding player is the ratio $\frac{v \cdot f(y)}{v \cdot y}$. By the minimax theorem (Theorem 20), the resulting zero-sum game has a minimax value achieved by the optimal mixed strategy $f(y)$ and the worst case v for $f(y)$. This value is thus precisely equal to the approximation ratio $\alpha_{\pi,\pi'}(y)$ of the optimal oblivious rounding scheme at y (recall Definition 10). Note that we require the rounding to be oblivious, hence the rounding player does not know the strategy v of the instance player when choosing her mixed strategy $f(y)$ given y.

Again by Theorem 20, the minimax value of the game $\alpha_{\pi,\pi'}(y)$ is alternatively achieved by first letting the instance player pick an optimal mixed strategy (a distribution $v \in C(V)$ over instances), and then allowing the rounding player to pick a best-response feasible solution $x \in X$. Notice that a mixed strategy v of the instance player is an instance of the *closure* $\mathrm{cl}(\pi)$ of the original problem π. Given y and v, the feasible solution x that maximizes the rounding player's expected payoff $\frac{v \cdot x}{v \cdot y}$ is precisely the same x that achieves the integrality gap $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(v, y)$ in Definition 4. Since the instance player is playing an optimal mixed strategy, we get that the value of the game $\alpha_{\pi,\pi'}(y)$ is equal to $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y)$. We conclude that the best approximation ratio at y and the integrality gap at y with respect to the closure coincide.

Given the above paragraphs, it may seem that the proof of Theorem 16 should follow directly by invoking Theorem 20. However, the classic minimax theorem is not immediately applicable in our setting due to a technical difficulty: While we can set the payoff function g in (2) to be $\frac{v \cdot x}{v \cdot y}$, the approximation ratio and integrality gap notions are defined with $\inf_{v \in V_y^+}$ instead of $\min_{v \in V}$ (to avoid division by zero). And while X and V_y^+ are bounded and X is also closed, V_y^+ may not be closed, and therefore may not be compact (unlike V). Lemma 33 and its proof show how to circumvent this problem by defining an appropriate series of zero-sum games. The lemma, its proof and the proof of Theorem 16 are deferred to Appendix B.1.

4 Application: Welfare Maximization

In this section we demonstrate our framework and results by applying them to the optimization problem of welfare maximization in combinatorial auctions. In Section 4.1 we state some preliminaries regarding the problem. In Section 4.2 we show a fairness property of oblivious rounding for welfare maximization. In Section 4.3 we bound the approximation ratio of oblivious rounding schemes for welfare maximization with unit demand valuations. In Section 4.4 we use the particular structure of the welfare maximization problem to extend our impossibility results to rounding of solutions that are guaranteed to be optimal (this is in contrast to the general case, see, e.g., Example 35). In Section 4.5 we give an explicit example of an instance that manages to "fool" oblivious rounding attempts.

4.1 Auction Preliminaries

A combinatorial auction involves a set $N = [n]$ of players and a set $M = [m]$ of indivisible items. Each player i has a valuation ν_i, which is a function $\nu_i : 2^M \to \mathbb{R}_{\geq 0}$ that assigns a real value to every subset of items $S \subseteq M$ (also called a *bundle*). Valuations are routinely assumed to be monotone (for every two bundles $S \subseteq T$, $\nu(S) \leq \nu(T)$), and bounded (assigning values up to some maximum value μ). An *allocation* $(S_1, \ldots, S_n)$ of the items is a (partial) partition of M into n bundles of items, one per player (some bundles may be empty). The *welfare* of a given allocation is the sum of the players' values for their allocated bundle, i.e., $\sum_{i=1}^n \nu_i(S_i)$. The goal of the welfare maximization problem is to find an allocation of the items that maximizes the welfare.

In the terminology of our framework, an instance of the welfare maximization problem is a vector v of dimension $n \cdot 2^m$ (indexed by pairs (i, S) of player and bundle) containing all the players' values for all the bundles, that is, $v_{i,S} = \nu_i(S)$. A feasible solution is a $\{0,1\}$-vector x of the same length, $n \cdot 2^m$, that indicates which player receives which bundle (up to one bundle per player), and does not over-allocate the items. Formally, $x_{i,S} \in \{0,1\}$, for every player i, $\sum_S x_{i,S} \leq 1$, and for every item j, $\sum_{i,S:j \in S} x_{i,S} \leq 1$.

The welfare maximization problem can be formulated as an integer program, and its standard relaxation is the associated linear program, called the *configuration LP* (see Appendix C.1). A relaxed solution is a vector y with $[0,1]$-entries, which can be thought of as an allocation of *fractional* rather than indivisible items, via an allocation of fractions of bundles. It must still hold that at most one of each item is allocated ($\sum_{i,S:j \in S} y_{i,S} \leq 1$ for every item j), and that each player receives at most one bundle ($\sum_S y_{i,S} \leq 1$ for every player i). In other words, a relaxed solution is any (fractional) feasible solution to the configuration LP.

A class of welfare maximization problems that has been extensively studied in the literature is welfare maximization with *gross substitutes* valuations. Such valuations play a crucial role in microeconomics [14] and in discrete convex optimization [19]; for a recent algorithmic survey see [21]. There are many equivalent definitions of gross substitutes valuations, one of which we give for completeness in Appendix C.1.

An important property of gross substitutes valuations is that the integrality gap of the configuration LP is 1.

▶ **Proposition 21** ([2]). *The integrality gap of the configuration LP for gross substitutes valuations is 1.*

Moreover, if all valuations are gross substitutes, then the welfare maximization problem can be solved optimally in polynomial time [17, 18].

A subclass of gross substitutes valuations is the class of *unit-demand* valuations. A valuation ν is unit-demand if there exists a vector $(\nu_1, \dots, \nu_m) \in \mathbb{R}^m_{\geq 0}$ such that for every bundle S, $\nu(S) = \max_{j \in S}(\nu_j)$.

Also relevant to our study is the class of *coverage* valuations. A valuation ν is a coverage function if it can be described by a tuple $\nu = \langle E, w, \{E_j\}_j \rangle$, where: (1) E is a ground set of elements, (2) $w : E \to \mathbb{R}_{\geq 0}$ is a weight function that assigns a weight $w(e)$ for every element $e \in E$, and (3) for every item $j \in [m]$, $E_j \subseteq E$ is the subset of elements *covered* by item j; and for every bundle of items $S \subseteq M$, it holds that $\nu(S) = \sum_{e \in \bigcup_{j \in S} E_j} w(e)$. The class of coverage valuations is a strict superset of unit-demand valuations (and is incomparable with gross substitutes). Coverage valuations are well-studied, with a particular surge in attention in the context of social networks (see, e.g., [1, 5]).

The convex hull of the class of unit-demand valuations is strictly larger than the class itself. In particular, the following lemma asserts that the convex hull of unit-demand valuations is precisely the class of coverage valuations (see Appendix C.2 for a proof).

▶ **Lemma 22.** *The class of coverage valuations is the convex hull of unit-demand valuations.*

4.2 A Fairness Property

In the context of welfare maximization, oblivious rounding with good approximation guarantees also offers certain guarantees *per player*. The intuition is that a rounding scheme that is ignorant to the instance has no way of telling which player contributes what to the welfare, and so must approximately preserve the welfare contributions of *all* players from behind its veil of ignorance. This can be viewed as a fairness property of oblivious rounding.

▶ **Proposition 23.** *Consider an oblivious rounding scheme f for the welfare maximization problem and its configuration LP relaxation, which has approximation ratio α. Then for every instance v and fractional allocation y, $f(y)$ guarantees for each player i, in expectation, an α-fraction of the player's value $\sum_S v_{i,S} y_{i,S}$ in y.*

Proof. Assume for contradiction that there is a player i for which this is not the case. Then we can create a new instance v' in which only player i's valuation is non-zero, meaning that all welfare comes from this player (note that while we do not allow an all zero valuation, assigning zero valuations to all players other than player i is valid). Since f is oblivious, it should achieve the approximation ratio α for v', contradiction. ◀

4.3 Impossibility Results

In this section we prove two impossibility results on the approximation ratios of oblivious rounding schemes for unit-demand valuations. These bounds extend to gross substitutes valuations.

▶ **Proposition 24.** *The approximation ratio of any oblivious rounding scheme for welfare maximization with two unit-demand players and the configuration LP relaxation is at most* $5/6$.

▶ **Proposition 25.** *The approximation ratio of any oblivious rounding scheme for welfare maximization with* n *unit-demand players and the configuration LP relaxation is at most* ≈ 0.782.

These impossibility results are in stark contrast to Proposition 21. In particular, while the integrality gap of the configuration LP is 1 even for a strict superclass of unit demand (i.e., gross substitutes), oblivious rounding for unit-demand valuations is quite limited in its performance.

Proof of Proposition 24. By Corollary 18 and Lemma 22, it is sufficient to show an instance with two coverage valuations that has an integrality gap of $5/6$. We claim that the instance in [10] for two players with submodular valuations satisfies these conditions. Let us describe the example explicitly using our notation, and showing in the process that the players' valuations are coverage functions.

There are four items and two players. For reasons that will become apparent shortly, it will be convenient to name the items $a_{11}, a_{12}, a_{21}, a_{22}$. There are six elements $\{H_1, H_2, V_1, V_2, D_1, D_2\}$. In both valuation functions, the coverage of elements by items is identical, but they differ in the weights of the different elements. We first state the coverage structure. For every element e, we denote the set of items that cover element e by $\bar{e}$. Let $\bar{H_1} = \{a_{11}, a_{12}\}$, $\bar{H_2} = \{a_{21}, a_{22}\}$, $\bar{V_1} = \{a_{11}, a_{21}\}$, Similarly, let $\bar{V_2} = \{a_{12}, a_{22}\}$, $\bar{D_1} = \{a_{11}, a_{22}\}$, and $\bar{D_2} = \{a_{12}, a_{21}\}$.

We now state the weights of the elements according to ν_1 and ν_2. Let $w^i(e)$ denote the weight of element e according to ν_i. For player 1, $w^1(H_1) = w^1(H_2) = 0$, $w^1(V_1) = w^1(V_2) = 2$, and $w^1(D_1) = w^1(D_2) = 1$. For player 2, $w^2(H_1) = w^2(H_2) = 2$, $w^2(V_1) = w^1(V_2) = 0$, and $w^2(D_1) = w^2(D_2) = 1$.

For example, $\nu_1(\{a_{11}, a_{12}\}) = w^1(H_1) + w^1(V_1) + w^1(V_2) + w^1(D_1) + w^1(D_2) = 6$, and $\nu_2(\{a_{11}, a_{12}\}) = w^2(H_1) + w^2(V_1) + w^2(V_2) + w^2(D_1) + w^2(D_2) = 4$.

One may verify that the following fractional solution has welfare 12: player 1 receives a fraction $1/2$ of bundle $\bar{H_1}$ and a fraction $1/2$ of bundle $\bar{H_2}$. This gives player 1 value 6. Player 2 receives a fraction $1/2$ of bundle $\bar{V_1}$ and a fraction $1/2$ of bundle $\bar{V_2}$. This gives player 2 value 6. It can be verified that no integer assignment of items gives total welfare above 10, establishing that the integrality gap is no better than $5/6$. This establishes the assertion of the proposition. ◀

Proof of Proposition 25. By Corollary 18 and Lemma 22, it is sufficient to show an instance with n coverage valuations and an integrality gap of ≈ 0.782. We claim that the instance in [10] for n players with submodular valuations satisfies these conditions.

Let us recall the instance. There are n players and n^n items arranged in an n dimensional cube. A *line* in direction i is a set of n points whose projection on the ith coordinate gives all values from 0 to $n-1$. There are $n^{n(n-1)}$ lines in direction i. The valuation function ν_i is defined such that $\nu_i(S)$ equals the fraction of lines in direction i hit by set S. One can verify that the valuation of player i is the following coverage valuation: Associate an element with every line in direction i, and let each item cover the elements corresponding to lines that contain it. The weight of every element is $1/n^{n(n-1)}$. As shown in [10], the integrality gap of this instance is ≈ 0.782. This establishes the assertion of the proposition. ◀

4.4 Impossibility Results for Optimal Solutions

We now define a strong notion of per-player guarantee. Consider an instance of the welfare maximization problem. A relaxed solution $y = \{y_{i,S}\}$ (a fractional solution of the configuration LP) is said to be *individually optimal* for this instance if the fractional value of every player in the solution x is his maximum possible value. Assuming that all valuations are monotone, this means that $\sum_S y_{i,S}\nu_i(S) = \nu_i(M)$ for every player i.

The significance of individual optimality lies in the following lemma. Consider a class of valuations P. Let v be an instance with valuations $\nu_1, \ldots, \nu_n \in C(P)$, and let $\rho(v)$ denote its integrality gap (as defined in Definition 4; we omit here the problem and relaxation from the notation). Let $y = \{y_{i,S}\}$ be an *optimal* relaxed solution for instance v, i.e., an optimal fractional solution to the configuration LP, whose welfare (LP objective value) we denote by $\mathrm{LP}(v, y)$.

▶ **Lemma 26.** *If y is individually optimal for v, then the approximation ratio for optimal solutions of any oblivious rounding scheme at y is at most $\rho(v)$.*

Proof. By definition, for every i, there exist valuations $\nu_{ik} \in P$ such that $\nu_i = \sum_k \lambda_{ik}\nu_{ik}$. For valuation ν_i and a fractional solution y, let $\nu_i(y) = \sum_S y_{i,S}\nu_i(S)$. Let $\alpha^*(y)$ denote the approximation ratio for optimal solutions of an oblivious rounding of y with respect to P, and let f be the oblivious rounding scheme achieving $\alpha^*(y)$. Consider random instances with valuations in P, where in every instance player i has valuation ν_{ik} with probability λ_{ik} (independently). For every random instance, the expected welfare obtained by f is at least $\alpha^*(y) \cdot \sum_i \nu_{ik}(y)$ (Proposition 23).

We claim that the individual optimality of y implies that y is also individually optimal for every random instance (i.e., $\nu_{ik}(y) = \nu_{ik}(M)$ for every i, k). Suppose otherwise, i.e., suppose there exist i, k such that $\nu_{ik}(y) < \nu_{ik}(M)$. Then, for that player i it follows (by monotonicity of the valuation) that $\sum_k \lambda_{ik}\nu_{ik}(y) < \sum_k \lambda_{ik}\nu_{ij}(M)$. On the other hand, $\sum_k \lambda_{ik}\nu_{ik}(y) = \nu_i(y) = \nu_i(M)$, so we get $\nu_i(M) > \sum_k \lambda_{ik}\nu_{ik}(M)$, contradiction.

Substituting $\nu_{ik}(y) = \nu_{ij}(M)$, and taking a weighted average over all instances, we get that the expected value obtained by f is at least $\alpha^*(y) \sum_{ik} \lambda_{ik}\nu_{ik}(M) = \alpha^*(y) \sum_i \nu_i(M) = \alpha^*(y)\,\mathrm{LP}(v, y)$. Now observe that f obtains the same ratio $\alpha^*(y)$ on the original instance v; therefore, $\alpha(y) \leq \rho(v)$ (otherwise, it contradicts the integrality gap of v). ◀

The following proposition follows directly from Lemma 26.

▶ **Proposition 27.** *Consider the problem of welfare maximization with valuations from P and its configuration LP relaxation. Let v be an instance attaining the integrality gap for $C(P)$, and let $y = \{y_{i,S}\}$ be an optimal solution of the configuration LP for instance v. If y is individually optimal, then the approximation ratio for optimal solutions α^* of any oblivious rounding scheme is at most the integrality gap of $C(P)$.*

Proof. Follows directly from Lemma 26, and from the definition of the approximation ratio for optimal solutions. Recall that this ratio is the infimum over the approximation ratios for optimal solutions of all $y \in Y$ for which V_y^* is nonempty (Definition 14). ◀

▶ **Corollary 28.** *The impossibility results in Propositions 24 and 25 apply also to the approximation ratio of oblivious rounding of optimal solutions of the configuration LP.*

Proof. The proof is by applying Proposition 27 and verifying that the instances in the proofs of Propositions 24 and 25 admit an individually optimal fractional solution. In the instance used in the proof of Proposition 24, the optimal fractional solution is individually optimal since this solution gives agent 1 a fraction 1/2 of each of $\bar{H}_1, \bar{H}_2$, and agent 2 a fraction 1/2 of each of $\bar{V}_1, \bar{V}_2$. In the instance used in the proof of Proposition 25, the optimal fractional solution is individually optimal since the solution gives a player i the n level sets with respect to coordinate i, each with probability $1/n$. ◀

4.5 How to Fool Oblivious Rounding

To gain intuition as to why oblivious rounding fails to round optimally, we now describe a two-player instance related to the instance in the proof of Proposition 24, and show why no oblivious rounding can succeed in rounding it with an approximation ratio better than 5/6. The instance is simple, including two players with unit-demand valuations and $\{0, 1\}$ values.

▶ **Example 29.** There are four items and two players. The items are $a_{11}, a_{12}, a_{21}, a_{22}$. Recall that a unit-demand function ν_i can be expressed by $\{\nu_{ij}\}_{j \in M}$, where $\nu(S) = \max_{j \in S} \nu_{ij}$. In our example, $\nu_{ij} \in \{0, 1\}$ for every i, j. We adopt the following notation used in the proof of Proposition 24: $\bar{H}_1 = \{a_{11}, a_{12}\}$, $\bar{H}_2 = \{a_{21}, a_{22}\}$, $\bar{V}_1 = \{a_{11}, a_{21}\}$, $\bar{V}_2 = \{a_{12}, a_{22}\}$, $\bar{D}_1 = \{a_{11}, a_{22}\}$, and $\bar{D}_2 = \{a_{12}, a_{21}\}$. We denote by S^i the items j such that $\nu_{ij} = 1$. The valuation functions are as follows: S^1 is $\bar{V}_1$ or $\bar{V}_2$, each with probability 1/3, and is $\bar{D}_1$ or $\bar{D}_2$, each with probability 1/6. S^2 is $\bar{H}_1$ or $\bar{H}_2$, each with probability 1/3, and is $\bar{D}_1$ or $\bar{D}_2$, each with probability 1/6.

Observe that for every realization of the valuations there exists an integral solution with social welfare 2. In addition, for every realization it holds that $\nu_1(\bar{H}_1) = \nu_1(\bar{H}_2) = 1$ and $\nu_2(\bar{V}_1) = \nu_2(\bar{V}_2) = 1$. Therefore, a fractional solution that assigns a fraction 1/2 of each of $\bar{H}_1$ or $\bar{H}_2$ to player 1, and a fraction 1/2 of each of $\bar{V}_1$ or $\bar{V}_2$ to player 2, obtains optimal welfare of 2.

We next show that for every integral solution the expected social welfare is at most 5/3. Assigning $\bar{H}_1$ to player 1 and $\bar{H}_2$ to player 2, or vice versa, grants player 1 value 1 and player 2 an expected value of 2/3. An analogous argument holds for the assignment of $\bar{V}_1$ and $\bar{V}_2$; and the assignment of $\bar{D}_1$ and $\bar{D}_2$ grants every player an expected value of 5/6. Each of these assignments gives welfare 5/3. Finally, it is easy to see that assigning a single item to one player and a triplet to the other derives even less welfare (3/2).

We conclude that any oblivious rounding obtains welfare at most 5/3, which is 5/6 of the optimal solution.

5 Oblivious Rounding in the Literature

We list here several examples of rounding schemes which are oblivious, as well as schemes which are not oblivious. It is interesting to notice that for welfare maximization with budget additive valuations, which is not closed under convex combinations, the best known approximation is not oblivious, whereas for welfare maximization with submodular valuations, which is closed under convex combinations, the best-known approximation is oblivious. Additional examples appear in [7].

Examples of Oblivious Rounding

- Threshold rounding for vertex cover [13].
- Randomized rounding for set cover [22].
- Random hyperplane rounding for max cut [11].
- Welfare maximization for fractionally subadditive (XOS) and submodular valuations [8, 10].
- Randomized metarounding for congestion [3].

Examples of Non-Oblivious Rounding

- Rounding of semidefinite programs (SDPs) for the constraint satisfaction problem (CSP) [23].
- Welfare maximization with budget-additive valuations [25, 4].
- Facility location [16].

6 Conclusion and Open Questions

In this work we have systematically studied the notion of oblivious rounding and its approximation guarantees, with applications to the welfare maximization problem. We mention several directions for future research. First, are there optimization problems that are not closed under convex combinations, where the best known approximation is achieved by an oblivious rounding scheme, and can potentially be improved by considering non-oblivious rounding schemes? For which problems are there polynomial-time computable oblivious rounding schemes that are comparable to the integrality gap? Finally, what else can we hope to learn about the most promising rounding techniques from properties of the combinatorial problem?

References

1 Ashwinkumar Badanidiyuru, Shahar Dobzinski, Hu Fu, Robert Kleinberg, Noam Nisan, and Tim Roughgarden. Sketching valuation functions. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1025–1035, 2012.

2 Sushil Bikhchandani and John W. Mamer. Competitive equilibrium in an exchange economy with indivisibilities. *Journal of Economic Theory*, 74(2):385–413, 1997.

3 Robert Carr and Santosh Vempala. Randomized metarounding. *Random Structures and Algorithms*, 20(3):343–352, 2002.

4 Deeparnab Chakrabarty and Gagan Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP. *SIAM J. Comput.*, 39(6):2189–2211, 2010.

5 Nan Du, Yingyu Liang, Maria-Florina Balcan, and Le Song. Learning time-varying coverage functions. In *Proceedings of the 27th Neural Information Processing Systems Conference*, pages 3374–3382, 2014.

6 Shaddin Dughmi, Tim Roughgarden, and Qiqi Yan. From convex optimization to randomized mechanisms: Toward optimal combinatorial auctions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, pages 149–158, 2011.

7 Paul Dütting, Thomas Kesselheim, and Éva Tardos. Algorithms as mechanisms: The price of anarchy of relax-and-round. In *Proceedings of the 16th ACM Conference on Economics and Computation*, pages 187–201, 2015.

8 Uriel Feige. On maximizing welfare when utility functions are subadditive. *SIAM J. Comput.*, 39(1):122–142, 2009.

9 Uriel Feige and Shlomo Jozeph. Oblivious algorithms for the maximum directed cut problem. *Algorithmica*, 71(2):409–428, 2015.
10 Uriel Feige and Jan Vondrák. The submodular welfare problem with demand queries. *Theory of Computing*, 6(1):247–290, 2010.
11 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.
12 Avinatan Hassidim and Yaron Singer. Submodular optimization under noise. Manuscript, 2016.
13 Dorit S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555—556, 1982.
14 Alexander S. Kelso and Vincent P. Crawford. Job matching, coalition formation, and gross substitutes. *Econometrica*, 50(6):1483–1504, 1982.
15 Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. *Journal of the ACM*, 58(6), 2011. Article 25.
16 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.
17 Kazuo Murota. Valuated matroid intersection I: Optimality criteria. *SIAM J. Discrete Math.*, 9(4):545–561, 1996.
18 Kazuo Murota. Valuated matroid intersection II: Algorithms. *SIAM J. Discrete Math.*, 9(4):562–576, 1996.
19 Kazuo Murota. *Discrete Convex Analysis.* Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2003.
20 Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129:192–224, 2006.
21 Renato Paes Leme. Gross substitutability: An algorithmic survey. Working paper, 2014.
22 Prabhakar Raghavan and Clark D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.
23 Prasad Raghavendra and David Steurer. How to round any CSP. In *Proceedings of the 50th Symposium on Foundations of Computer Science*, pages 586–594, 2009.
24 Tim Roughgarden, Inbal Talgam-Cohen, and Jan Vondràk. When are welfare guarantees robust? Working paper, 2016.
25 Aravind Srinivasan. Budgeted allocations in the full-information setting. In *Proceedings of the 11th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 247–253, 2008.
26 John von Neumann. Zur theorie der gesellschaftsspiele. *Math. Annalen.*, 100:295–320, 1928.
27 Neal E. Young. Randomized rounding without solving the linear program. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 170–178, 1995.

A Appendix for Section 2

A.1 Missing Proofs

Proof of Observation 5. Since X is compact, there exist $x^* \in X$ and $v^* \in V$ such that $x^* = \arg\max_{x \in X} v^* \cdot x$ (and thus in particular $v^* \in V^+_{x^*}$). Set the relaxed solution y to be x^* (this is a valid choice since $x^* \in Y$). Then $\rho_{\pi,\pi'}(y) \leq 1$, since 1 can be achieved by choosing the instance v in the definition of $\rho_{\pi,\pi'}(y)$ to be $v = v^*$. The observation follows. ◀

Proof of Observation 6. For every $y \in Y \setminus \{0^d\}$, the set of instances v such that $v \cdot y > 0$ expands when we replace $v \in V$ by $v \in C(V)$. Thus for every y, $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y) \leq \rho_{\pi,\pi'}(y)$, and the observation follows. ◀

Proof of Observation 7. Let $y \in Y \setminus \partial(Y)$ be a point not on the boundary, and let $\delta > 1$ be a scalar such that $\delta y \in \partial(Y)$. For every $v \in V_y^+$, $v \cdot y \leq v \cdot \delta y$, and so $\rho_{\pi,\pi'}(v, y) \geq \rho_{\pi,\pi'}(v, \delta y)$. It follows that $\rho_{\pi,\pi'}(y) \geq \rho_{\pi,\pi'}(\delta y)$, proving the observation. ◀

Proof of Observation 11. This follows from Definitions 4 (integrality gap) and 10 (approximation ratio), and by noticing that even if $f(y) \in C(X) \setminus X$, for every $v \in V$ there must be some $x \in X$ with $v \cdot x \geq v \cdot f(y)$. ◀

Proof of Observation 30. Fix y and $v \in V_y^+$. Let f' be the oblivious rounding scheme that rounds y to $f(\delta y)$. So f' is reasonable by definition, and achieves $\frac{v \cdot f'(y)}{v \cdot y} = \frac{v \cdot f(\delta y)}{v \cdot y} \geq \frac{v \cdot f(\delta y)}{v \cdot \delta y}$. Taking the infimum over v maintains these relations, and so the observation holds. ◀

Proof of Observation 31. Let $y \in Y \setminus \partial(Y)$, $y \neq 0^d$ be a point not on the boundary, and let $\delta > 1$ be a scalar such that $\delta y \in \partial(Y)$. Since f is reasonable, $\alpha_{\pi,\pi'}(y) \geq \alpha_{\pi,\pi'}(\delta y)$, and since $\alpha_{\pi,\pi'}$ is achieved by taking the infimum over $Y \setminus \{0^d\}$, the observation follows. ◀

Proof of Observation 13. Let $y \in Y \setminus \partial(Y)$ be a point not on the boundary, and let $\delta > 1$ be a scalar such that $\delta y \in \partial(Y)$. Then for every $v \in V$ such that $v \cdot y > 0$, $v \cdot y < v \cdot \delta y$ and so $v \notin V_y^*$. If $v \cdot y = 0$, then by positivity of π', again $v \notin V_y^*$. We conclude that V_y^* is empty, completing the proof. ◀

A.1.1 Approximation Ratio of Reasonable Oblivious Rounding

We recall our assumption that π' is a convex relaxation. We say that an oblivious rounding scheme f is *reasonable* if it guarantees, for every relaxed solution $y \in Y \setminus \{0^d\}$, at least the approximation ratio $\alpha_{\pi,\pi'}(\delta y)$ that it achieves for $\delta y \in \partial(Y)$ (where $\delta \geq 1$ is the scaler by which y needs to be multiplied to reach the boundary). Assuming reasonability is without loss of generality as the following observation shows (see Appendix A for a proof):

▶ **Observation 30.** *For every oblivious rounding scheme f there is a reasonable oblivious rounding scheme f' such that for every $y \in Y' \setminus \{0^d\}$, the approximation ratio of f' at y is at least the approximation ratio of f at $\delta y \in \partial(Y)$, and so the overall approximation ratio of f' is at least that of f.*

For reasonable rounding schemes, the approximation ratios matter only on the boundary (see Appendix A for a proof):

▶ **Observation 31.** *The overall approximation ratio of any reasonable oblivious rounding scheme is not affected by the approximation ratios at relaxed solutions that lie strictly within the boundary: $\alpha_{\pi,\pi'} = \min_{y \in \partial(Y)} \alpha_{\pi,\pi'}(y)$.*

By Observations 7 and 31, if f is a reasonable oblivious rounding scheme and $\alpha_{\pi,\pi'}(y) = \rho_{\pi,\pi'}(y)$ for every $y \in \partial(Y)$, then f is tight (Definition 12).

B Appendix for Section 3

B.1 Proof of Theorem 16: Missing Details

We now formally state and prove our main lemma. We use $(C(V))_y^+$ to denote the set of instances $v \in C(V)$ such that $v \cdot y > 0$ (recall that V_y^+ is the set of such instances in V rather than in $C(V)$). We also use the following simple observation:

▶ **Observation 32.** *There exists $\epsilon > 0$ and $x' \in C(X)$ such that for every $v \in V$, $v \cdot x' \geq \epsilon$.*

Proof. Every feasible solution in X is a vector assigning nonnegative values to d variables $x_1, \ldots, x_d$. We may assume without loss of generality that for every coordinate $1 \leq j \leq d$, there is some solution $x^j \in X$ for which the variable x_j has strictly positive value. (Otherwise the variable x_j has value 0 in all feasible solutions and hence is redundant.) Consider a solution $x' = \frac{1}{d}\sum_{j=1}^{d} x^j \in C(X)$. All its coordinates are strictly positive. Recall that every $v \in V$ is nonnegative and not identically 0^d. Consequently $x' \cdot v > 0$ for every $v \in C(v)$. Moreover, our assumption that V is compact (see Section 2.1.1) together with the continuity of the inner product function implies that the function $f(v) = x' \cdot v$ attains a minimum over $v \in V$. Let $\epsilon = \min_{v \in V}[x' \cdot v]$ and note that $\epsilon > 0$. ◀

▶ **Lemma 33.** *Fix $y \in Y \setminus \{0^d\}$. There exists a value μ^* such that*

$$\mu^* = \max_{x \in C(X)} \inf_{v \in V_y^+} \frac{v \cdot x}{v \cdot y} = \inf_{v \in (C(V))_y^+} \max_{x \in X} \frac{v \cdot x}{v \cdot y}. \tag{3}$$

Moreover, there is a choice of $x^* \in C(X)$ and $v^* \in (C(V))_y^+$ such that x^* maximizes $\frac{v^* \cdot x}{v^* \cdot y}$, v^* minimizes $\frac{v \cdot x^*}{v \cdot y}$, and $\mu^* = \frac{v^* \cdot x^*}{v^* \cdot y}$.

Proof. Given $y \in Y \setminus \{0^d\}$, consider a series of two-player zero-sum games parameterized by $\mu \in \mathbb{R}_{\geq 0}$. In each such game, the *rounding* player has strategy set X, the *instance* player has strategy set V, and the payoff to the rounding player for choices $x \in X, v \in V$ is $v \cdot x - \mu(v \cdot y)$ (i.e., we use the difference as payoff instead of the ratio). Since X and V are both compact by assumption (see Section 2.1.1 and Remark 3.1), then Theorem 20 applies, and the unique minimax value p_μ of the game with parameter μ is

$$p_\mu = \max_{x \in C(X)} \min_{v \in V} \{v \cdot x - \mu(v \cdot y)\} = \min_{v \in C(V)} \max_{x \in X} \{v \cdot x - \mu(v \cdot y)\}.$$

Let $x_\mu \in C(X), v_\mu \in C(V)$ be equilibrium strategies that achieve the minimax value p_μ (by Theorem 20, such strategies are guaranteed to exist).

We now observe some properties of p_μ as a function of μ:

- p_μ is bounded: This is by the assumption that X and V are bounded.
- For sufficiently small μ, p_μ is positive: By Observation 32, $\max_{x \in c(X)} \min_{v \in V}\{v \cdot x\} \geq \min_{v \in C(V)}[x' \cdot v] \geq \epsilon$. Taking μ to be sufficiently small we can ensure that $\mu(v \cdot y) \leq \frac{\epsilon}{2}$ for every $v \in V$, because both y and V are bounded.
- For every μ such that $p_\mu \leq 0$ we have that $v_\mu \cdot y > 0$ for every equilibrium strategy v_μ (otherwise $p_\mu = v_\mu \cdot x_\mu$ and the rounding player can choose $x_\mu \in C(X)$ such that $v_\mu \cdot x_\mu > 0$). Hence $v_\mu \in (C(V))_y^+$.
- For large enough μ, p_μ is negative: Fix $v^+ \in C(V)_y^+$. Since $C(X)$ is bounded, we can set $\mu > (v^+ \cdot x)/(v^+ \cdot y)$ for every $x \in C(X)$. In particular, $v^+ \cdot x_\mu - \mu(v^+ \cdot y) < 0$, and so since v_μ is an equilibrium strategy, $p_\mu = v_\mu \cdot x_\mu - \mu(v_\mu \cdot y) \leq v^+ \cdot x_\mu - \mu(v^+ \cdot y) < 0$.
- p_μ is monotone (weakly) decreasing in μ: Let $\bar{\mu} > \mu$. Then
$$\begin{aligned} p_{\bar{\mu}} &= v_{\bar{\mu}} \cdot x_{\bar{\mu}} - \bar{\mu}(v_{\bar{\mu}} \cdot y) \\ &\leq v_\mu \cdot x_{\bar{\mu}} - \bar{\mu}(v_\mu \cdot y) \qquad (4) \\ &\leq v_\mu \cdot x_\mu - \mu(v_\mu \cdot y), \qquad (5) \end{aligned}$$
where (4) holds since $v_{\bar{\mu}}, x_{\bar{\mu}}$ are equilibrium strategies, and (5) holds since v_μ, x_μ are equilibrium strategies and $-\bar{\mu} < -\mu$.
- Let μ' be the smallest μ such that $p_{\mu'} \leq 0$, then p_μ is monotone *strictly* decreasing for $\mu \geq \mu'$: For every $\mu \geq \mu'$, by monotonicity $p_\mu \leq 0$, and so $v_\mu \cdot y > 0$. Thus for every $\mu \geq \mu'$, we can replace "$\leq$" by "$<$" in (5).
- p_μ is continuous when $\mu > 0$.

Given the above properties of p_μ, there is a unique $\mu^* > 0$ for which $p_{\mu^*} = 0$. We know that $v_{\mu^*} \cdot y > 0$, or equivalently, $v_{\mu^*} \in (C(V))^+_y$. The condition $v_{\mu^*} \cdot x_{\mu^*} - \mu^*(v_{\mu^*} \cdot y) = 0$ with positive $v_{\mu^*} \cdot y$ implies that $\mu^* = \frac{v^* \cdot x^*}{v^* \cdot y}$. This completes the proof. ◀

Proof of Theorem 16. Fix $y \in Y \setminus \{0^d\}$. On the one hand, for every oblivious rounding scheme f, recall from Definition 10 that the approximation ratio of f at y is $\alpha_{\pi,\pi'}(y) = \inf_{v \in V^+_y} \frac{v \cdot f(y)}{v \cdot y}$. Hence the oblivious rounding scheme with the optimal approximation ratio at y is the one that rounds y to $f(y) = \arg\max_{x \in C(X)} \inf_{v \in V^+_y} \frac{v \cdot x}{v \cdot y}$, achieving an approximation ratio of

$$\alpha_{\pi,\pi'}(y) = \max_{x \in C(X)} \inf_{v \in V^+_y} \frac{v \cdot x}{v \cdot y}. \tag{6}$$

On the other hand, recall from Definition 4 that the integrality gap at y with respect to the closure is

$$\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y) = \inf_{v \in (C(V))^+_y} \max_{x \in X} \frac{v \cdot x}{v \cdot y}. \tag{7}$$

So both parts of the theorem follow from Lemma 33, which states that (6) and (7) are equal. ◀

B.2 Rounding Optimal Solutions

A corollary of Theorem 16 applies to the approximation guarantees of oblivious rounding for solutions known to be optimal. The corollary follows directly from the observation in Section 2.2.2 that for every $y \in Y$ with nonempty V^*_y, $\alpha_{\pi,\pi'}(y) \le \alpha^*_{\pi,\pi'}(y)$.

▶ **Corollary 34.** *Given a problem $\pi = (V, X)$ and a relaxation $\pi' = (V, Y)$, there exists an oblivious rounding scheme $f : Y \to C(X)$ that achieves an approximation ratio of $\alpha^*_{\pi,\pi'}(y) \ge \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y)$ at every point y with nonempty V^*_y. The overall approximation ratio of f is $\alpha^*_{\pi,\pi'} \ge \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}$.*

If $\pi = \mathrm{cl}(\pi)$ then there exists an oblivious rounding scheme $f : Y \to C(X)$ that is individually tight for optimal solutions.

Proof. By Definitions 4 (integrality gap) and 14 (approximation ratio for optimal solutions), if for an oblivious rounding scheme f it holds that $\alpha_{\pi,\pi'}(y) = \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y)$ for every $y \in Y \setminus \{0^d\}$, then $\alpha^*_{\pi,\pi'}(y) \ge \alpha_{\pi,\pi'}(y) = \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y)$ for every y with nonempty V^*_y. By Theorem 16 there exists such an oblivious rounding scheme. It follows that $\alpha^*_{\pi,\pi'} \ge \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}$. If $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y) = \rho_{\pi,\pi'}(y)$ for every y with nonempty V^*_y, then $\alpha^*_{\pi,\pi'}(y) \ge= \rho_{\pi,\pi'}(y)$, which by definition implies individual tightness for optimal solutions (Definition 15). ◀

The next example shows that, unlike the case in Corollary 18, there may be oblivious rounding schemes whose approximation ratio for optimal solutions $\alpha^*_{\pi,\pi'}$ surpasses the integrality gap of the closure $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}$. The reason for this difference is that $\alpha^*_{\pi,\pi'}$ only takes into account relaxed solutions that are guaranteed to be optimal for some instance of the relaxation.

▶ **Example 35.** Consider a problem $\pi = (V, X)$ of dimension 2, where the instances are $V = \{v_1, v_2\} = \{(1, 0), (0, 1)\}$ and the feasible solutions are $X = \{x_1, x_2, x_3\} = \{(0, 0), (1, 0), (0, 1)\}$. (For concreteness this example can be thought of as a welfare maximization problem with a single item and two buyers, where either: the first buyer has value 1 for the item and the other has value 0 – this is the first instance; or vice versa – this is the second instance. See Section 4 for more on welfare maximization.) Consider a relaxation $\pi' = (V, Y)$ where Y is a quadrilateral "kite" with vertices $\{(0,0),(1,0),(\frac{3}{4},\frac{3}{4}),(0,1)\}$. The closures $\mathrm{cl}(\pi), \mathrm{cl}(\pi')$ have an instance set $C(V)$ which is the set of vectors $\{(\lambda, 1-\lambda) \mid \lambda \in [0,1]\}$.

Oblivious rounding of the point $y = (\frac{3}{4},\frac{3}{4})$ gives a point $f(y)$ that belongs to $C(X)$, i.e., to the triangle with vertices $\{(0,0),(1,0),(0,1)\}$. For any such point $f(y)$, $\min\{v_1 \cdot f(y), v_2 \cdot f(y)\} \le \frac{1}{2}$ whereas $v_1 \cdot y = v_2 \cdot y = \frac{3}{4}$, and so the approximation ratio $\alpha_{\pi,\pi'}(y)$ of any oblivious rounding scheme at y is $\le \frac{1}{2}/\frac{3}{4} = \frac{2}{3}$. By rounding y to $(\frac{1}{2},\frac{1}{2})$ we get $\alpha_{\pi,\pi'}(y) = \frac{2}{3}$. It also follows that the overall approximation ratio of the best oblivious rounding scheme is $\le \frac{2}{3}$.

Consider now the integrality gap $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y)$ at y with respect to the closure. For every $(\lambda, 1-\lambda) \in C(V)$, $\max\{(\frac{1}{2},\frac{1}{2}) \cdot x_1, (\frac{1}{2},\frac{1}{2}) \cdot x_2, (\frac{1}{2},\frac{1}{2}) \cdot x_3\} = \max\{\lambda, 1-\lambda\} \ge \frac{1}{2}$ whereas $(\lambda, 1-\lambda) \cdot y = \frac{3}{4}$, and so the integrality gap is $\le \frac{2}{3}$. Since $(\frac{1}{2},\frac{1}{2}) \in C(V)$ we get $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}(y) = \frac{2}{3}$. This is equal to the approximation ratio $\alpha_{\pi,\pi'}(y)$ of the best oblivious rounding scheme at y, as known from Theorem 16. It also follows that the overall integrality gap $\rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')}$ is $\le \frac{2}{3}$.

However, the point $y = (\frac{3}{4},\frac{3}{4})$ is not an optimal solution of the relaxation with respect to either of the instances in V. The set of optimal solutions $\{y \in Y \mid V^*_y \neq \emptyset\}$ includes only x_2 and x_3, and so the identity function is an oblivious rounding scheme with approximation ratio of 1 for optimal solutions. We conclude that $1 = \alpha^*_{\pi,\pi'} > \rho_{\mathrm{cl}(\pi),\mathrm{cl}(\pi')} = \frac{2}{3}$.

C Appendix for Section 4

C.1 Gross Substitutes and the Configuration LP

▶ **Definition 36.** A valuation ν is *gross substitutes* if the following holds. Consider any two item-price vectors $p, q \in \mathbb{R}^m$ such that $q \ge p$. Let S be a bundle such that $\nu(S) - \sum_{j \in S} p_j \ge \nu(T) - \sum_{j \in T} p_j$ for every bundle T. Let $S' = \{j \in S \mid q_j = p_j\}$. Then there exists a bundle U such that $S' \subseteq U$ and $\nu(U) - \sum_{j \in U} q_j \ge \nu(T) - \sum_{j \in T} q_j$ for every bundle T.

In words, a valuation is gross substitutes if for every bundle that maximizes the player's utility (value for the bundle minus the aggregate price of its items) given a price vector p, when prices of some of the items are raised, the items whose prices were not raised still participate in a bundle that maximizes the player's utility given the new price vector q. Intuitively, this monotonicity property facilitates the greedy approach in a similar way to matroid properties.

▶ **Definition 37.** The *integer programming* formulation of the welfare maximization problem is the following:

$$\max \textstyle\sum_{i,S} x_{i,S} v_{i,S}$$

s. t.

$$\sum_S x_{i,S} \le 1 \qquad \forall i \in N \tag{8}$$

$$\sum_{i,S:j\in S} x_{i,S} \le 1 \qquad \forall j \in M \tag{9}$$

$$x_{i,S} \in \{0,1\} \qquad \forall i \in N, S \subseteq M.$$

Constraint (8) corresponds to the requirement that no more than one bundle be allocated per player, and Constraint (9) corresponds to the requirement that no item is over-allocated. Note that the welfare maximization instance v appears only in the objective and does not affect X.

▶ **Definition 38.** The relaxed solution set Y of the *configuration LP relaxation* to the welfare maximization problem is the set of vectors y that are feasible solutions to the following LP:

$$\begin{aligned}
&\max \textstyle\sum_{i,S} y_{i,S} v_{i,S} \\
\text{s. t.}\quad & \\
&\textstyle\sum_S y_{i,S} \le 1 \qquad \forall i \in N \qquad (10)\\
&\textstyle\sum_{i,S:j\in S} y_{i,S} \le 1 \qquad \forall j \in M \qquad (11)\\
&y_{i,S} \ge 0 \qquad \forall i \in N, S \subseteq M.
\end{aligned}$$

Constraints (10) and (11) correspond to the same requirements as in the integer programming formulation above. However, the variables $y_{i,S}$ can now take any value in the interval $[0,1]$, unlike the integral constraint in the IP problem.

C.2 Proof of Lemma 22: Coverage is the Closure of Unit-Demand

In this section we provide a proof of Lemma 22 for completeness (*cf.*, [6], Appendix A.1).

Proof of Lemma 22. Let UD and COV be the classes of unit-demand and coverage valuations, respectively. To prove the proposition we show that $C(\text{UD}) \subseteq \text{COV}$ and $\text{COV} \subseteq C(\text{UD})$. To show that $C(\text{UD}) \subseteq \text{COV}$, it is shown, in Lemma 39, that every unit-demand valuation is a coverage valuation (i.e., $\text{UD} \subseteq \text{COV}$ and thus $C(\text{UD}) \subseteq C(\text{COV})$), and, in Lemma 40, we show that the $C(\text{COV}) \subseteq \text{COV}$. The fact that $\text{COV} \subseteq C(\text{UD})$ is established in Lemma 41, and this completes the proof. ◀

▶ **Lemma 39.** *Every unit-demand valuation is a coverage valuation.*

Proof. Let v be a unit-demand valuation. We describe a coverage valuation v' satisfying $v'(S) = v(S)$ for every set $S \subseteq M$. Assume, by renaming, that $v_1 \le v_2 \le \cdots \le v_m$, and let $\Delta_j = v_j - v_{j-1}$. Let D be the set of indices of distinct values; i.e., $D = \{j \in [m] | \Delta_j > 0\}$ (with the convention that $v_0 = -1$). Associate an element with every distinct value v_j and set its weight to Δ_j. For every item j, E_j (i.e., the set of elements covered by j) is the set of elements corresponding to items up to item j. For example, if there are 4 items with values $v_1 = 1, v_2 = 1, v_3 = 3, v_4 = 8$, then there would be three elements, corresponding to items $1, 3, 4$ with weights $1, 2, 5$, respectively. For every $S \subseteq M$ it holds that

$$v'(S) = \sum_{e \in \bigcup_{j\in S} E_j} w(e) = \max_{j\in S} \sum_{e\in E_j} w(e) = \max_{j\in S} \sum_{k\in D \wedge k \le j} \Delta_k = \max_{j\in S} v_j = v(S),$$

as desired. ◀

▶ **Lemma 40.** *A convex combination of coverage functions is a coverage function.*

Proof. Let $v^1 = \langle E^1, w^1, \{E^1_j\}_j\rangle$ and $v^2 = \langle E^2, w^2, \{E^2_j\}_j\rangle$ be two coverage functions. It is sufficient to show that for every $\lambda \in [0,1]$, $v(S) = \lambda v^1(S) + (1-\lambda)v^2(S)$ is a coverage function, where S ranges over all subsets of M. Let $E = E^1 \biguplus E^2$, and let $w : E \to R^{\ge 0}$ be a weight function defined as $w(e) = \lambda w^1(e)$ for every $e \in E^1$, and $w(e) = (1-\lambda)w^2(e)$ for every $e \in E^2$. Finally, for every item $j \in M$, let $E_j = E^1_j \biguplus E^2_j$. Consider the coverage function $v = \langle E, w, \{E_j\}_j \rangle$. For every set $S \subseteq M$ it holds that

$$\begin{aligned} v(S) &= \sum_{e \in \bigcup_{j \in S} E_j^1 \uplus E_j^2} w(e) \\ &= \sum_{e \in \bigcup_{j \in S} E_j^1} w(e) + \sum_{e \in \bigcup_{j \in S} E_j^2} w(e) \\ &= \sum_{e \in \bigcup_{j \in S} E_j^1} \lambda w^1(e) + \sum_{e \in \bigcup_{j \in S} E_j^2} (1-\lambda) w^2(e) \\ &= \lambda v^1(S) + (1-\lambda) v^2(S), \end{aligned}$$

as desired. ◀

▶ **Lemma 41.** *Every coverage valuation can be expressed as a convex combination of unit-demand valuations.*

Proof. Let $v = \langle E, w, \{E_j\}_j \rangle$ be a coverage valuation, and let $k = |E|$ be the number of elements in E. We show that there exist k unit-demand valuations, whose average valuation for any set S equals $v(S)$. Associate a unit-demand function with every element as follows. For every element $e \in E$, let $S_e = \{j \in M : e \in E_j\}$ be the set of items that cover element e. The unit-demand valuation v^e associated with element e is defined by

$$v_j^e = \begin{cases} k \cdot w(e), & \text{if } j \in S_e \\ 0, & \text{otherwise.} \end{cases}$$

For every set of items $S \subseteq M$, let $E_S = \bigcup_{j \in S} E_j$, and let $\mathbb{1}\{e \in E_S\}$ be a binary function that returns 1 iff $e \in E_S$. We show that $v(S)$ can be written as a convex combination of the unit-demand functions described above. Indeed, for every set $S \subseteq M$,

$$\begin{aligned} \frac{1}{k} \sum_{e \in E} v^e(S) =& \frac{1}{k} \sum_{e \in E} \mathbb{1}\{e \in E_S\} k \cdot w(e) \\ =& \frac{1}{k} \sum_{e \in E_S} k \cdot w(e) \\ =& \sum_{e \in E_S} w(e) = v(S). \end{aligned}$$

◀

A Deterministic Fully Polynomial Time Approximation Scheme for Counting Integer Knapsack Solutions Made Easy[*]

Nir Halman[†]

Hebrew University of Jerusalem, Israel
halman@huji.ac.il

Abstract

Given n elements with nonnegative integer weights $w = (w_1, \ldots, w_n)$, an integer capacity C and positive integer ranges $u = (u_1, \ldots, u_n)$, we consider the counting version of the classic integer knapsack problem: find the number of distinct multisets whose weights add up to at most C. We give a deterministic algorithm that estimates the number of solutions to within relative error ϵ in time polynomial in $n, \log U$ and $1/\epsilon$, where $U = \max_i u_i$. More precisely, our algorithm runs in $O(\frac{n^3 \log^2 U}{\epsilon} \log \frac{n \log U}{\epsilon})$ time. This is an improvement of n^2 and $1/\epsilon$ (up to log terms) over the best known deterministic algorithm by Gopalan *et al.* [FOCS, (2011), pp. 817-826]. Our algorithm is relatively simple, and its analysis is rather elementary. Our results are achieved by means of a careful formulation of the problem as a dynamic program, using the notion of *binding constraints*.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems (Computations on discrete structures), G.2.1 Combinatorics (Counting problems), I.2.8 Problem Solving, Control Methods, and Search (Dynamic programming)

Keywords and phrases Approximate counting, integer knapsack, dynamic programming, bounding constraints, K-approximating sets and functions.

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.9

1 Introduction

In this paper we target at designing a deterministic fully polynomial time approximation scheme (FPTAS) for one of the most basic #P-complete counting problems – counting the number of integer knapsack solutions. Given n elements with nonnegative integer weights $w = (w_1, \ldots, w_n)$, an integer capacity C, and positive integer ranges $u = (u_1, \ldots, u_n)$, we consider the counting version of the classic integer knapsack problem: find the size of the set of feasible solutions $\text{KNAP}(w, C, u) = \{x \mid \sum_{i \le n} w_i x_i \le C,\ 0 \le x_i \le u_i\}$. (We assume, w.l.o.g., that $w_i u_i \le C$ for all i.) We give a deterministic FPTAS for this problem that for any tolerance $\epsilon > 0$ estimates the number of solutions within relative error ϵ in time polynomial in the (binary) input size and $1/\epsilon$.

Our result. Our main result is the following theorem (the base of the logarithms in this paper are all 2 unless otherwise specified).

[*] A full version of this paper is available online in http://www.optimization-online.org/DB_HTML/2015/12/5231.html.

[†] Partial support for this research was provided by the Recanati Fund of the Jerusalem School of Business Administration.

▶ **Theorem 1.** *Given a knapsack instance $KNAP(w, C, u)$ with $U = \max_i u_i$ and $\epsilon > 0$, there is a deterministic $O(\frac{n^3 \log^2 U}{\epsilon} \log \frac{n \log U}{\epsilon})$ algorithm that computes an ϵ-relative error approximation for $|KNAP(w, C, u)|$.*

Relevance to existing literature. The field of approximate counting is largely based on Markov Chain Monte Carlo Sampling [6], a technique that is inherently randomized, and has had remarkable success, see [10] and the references therein. The first approximation schemes for counting integer knapsack solutions are fully polynomial *randomized* approximation schemes (FPRASs). Given parameters $\epsilon > 0$ for the error tolerance and $1 > \delta > 0$ for the failure probability, the FPRAS returns a solution which is correct with probability at least $1 - \delta$, and the running time is required to be polynomial in the (binary) input size, $1/\epsilon$ and in $\log(1/\delta)$. To the best of our knowledge, the best FPRAS up to date is given by Dyer [1], and is achieved by combining dynamic programming (DP, to be distinguished from dynamic program by context) with simple rejection sampling. The complexity of the algorithm is $O(n^5 + n^4/\epsilon^2)$, so in fact the algorithm is strongly polynomial (see, e.g., [7]), that is, the number of arithmetic operations is polynomial in n and independent of C, U.

To the best of our knowledge, the currently best (deterministic) FPTAS for this problem is given by Gopalan *et al.* [3], and has complexity $O(\frac{n^5}{\epsilon^2} \log^2 U \log W)$, where $W = \sum_i w_i u_i + C$ (see also [2]). We note that the real achievement of [2] is providing an FPTAS for the *multidimensional* version of the problem. Because of this reason they use a somewhat more sophisticated approach than ours, relying on read-once branching programs and insight from Meka and Zuckerman [8].

We note in passing that the first (deterministic) FPTAS for counting 0/1 knapsack solutions (i.e., our problem restricted to the case where $u = (1, \ldots, 1)$) is given by Štefankovič *et al.* [10] and runs in $O(n^3 \epsilon^{-1} \log(n/\epsilon))$ time. The currently best (deterministic) FPTAS runs in $O(n^3 \epsilon^{-1} \log(1/\epsilon)/\log n)$ time [9].

Technique used. In this paper we give two FPTASs that are based upon formulating the counting problem as a DP. Instead of deciding at once how many copies of item i to put in the knapsack, we split the decision into a sequence of at most $\log u_i$ binary sub-decisions concerning (not necessarily all the) bundles of $1, 2, 4, \ldots, 2^{\lfloor \log u_i \rfloor}$ copies of the item. In order to translate this into a DP, we use the idea of what we call *binding constraints*, as explained in detail below. The first FPTAS uses a primal DP formulation and approximates it via the recent technique of K-approximation sets and functions introduced by [5], which we overview in Section 3.1. The second FPTAS uses a dual DP formulation and approximates it in a similar way [10] approximate the 0/1 knapsack problem. We overview their solution in Section 4.1.

Our contribution. While not strongly polynomial, the running time of our solutions are of order n and $1/\epsilon$ (up to log terms) faster than the (randomized, but strongly-polynomial) algorithm of Dyer [1]. The complexity of our solutions is also better by factors of n^2 and $1/\epsilon$ (up to log terms) than the (non strongly-polynomial, but deterministic) algorithm of Gopalan *et al.* [3]. Moreover, our algorithms are relatively simple and their analysis is rather elementary. A second contribution is our new DP technique – *binding constraints*, which may be of independent interest.

Organization of the paper. In Section 2 we introduce the notion of binding constraints. In Section 3 we design an FPTAS which is based upon a primal DP formulation of the problem.

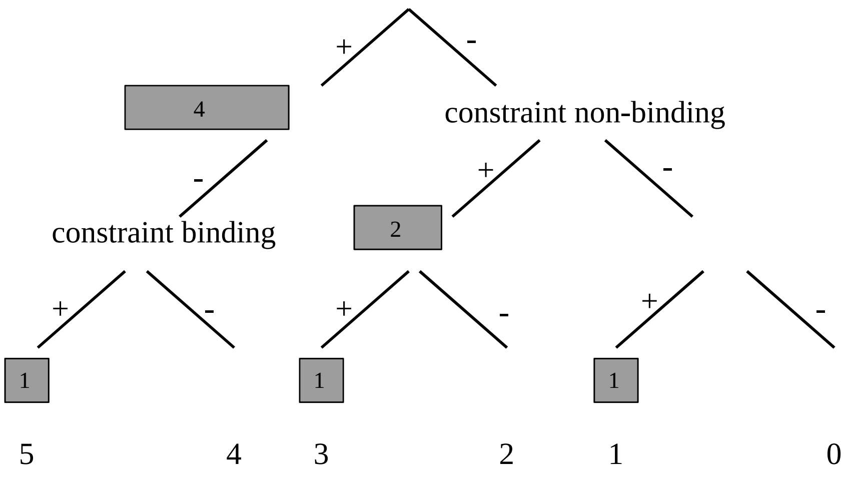


Figure 1 A list of feasible solutions for a knapsack of size 5 and a single item of weight 1 and maximal number of copies 5, i.e., $\mathrm{KNAP}((1), 5, (5))$.

Our second FPTAS, based upon a dual DP formulation, is given in Section 4. In this way we showcase that the notion of binding constraints is useful for the primal as well as the dual DP formulation.

2 Binding constraints and the integer knapsack problem

In this section we present the idea behind the notion of binding constraints. Instead of deciding at once how many copies of item i to put in the knapsack, we split the decision into $\lfloor \log u_i \rfloor + 1$ binary sub-decisions. If the values of the various u_i are all powers of 2 (minus one), then the binary sub-decisions $j = 1, \dots, \lfloor \log u_i \rfloor + 1$ for item i are equivalent to deciding whether to put in the knapsack "bundles" of 2^{j-1} copies of item i. E.g., for $u_1 = 7$ we split the decision concerning item 1 into the 3 independent binary sub-descisions of whether to put in the knapsack 1, 2, 4 more copies of item 1. In this case there is a simple DP formulation which is equivalent to a 0/1 knapsack problem with exactly $\sum_{i=1}^{n} \log(u_i + 1)$ items. But when not all values of the various u_i are powers of 2 (minus one), then splitting into binary sub-decisions is more complicated as a binary sub-decision may not be independent on the previous sub-decision. We demonstrate this in Example 2.

▶ **Example 2.** Suppose we have a knapsack of size 5 and at most 5 copies of a single item of weight 1, i.e., the instance $(w = (1);\ C = 5;\ u = (5))$. We split the item into 3 different subitems consisting of bundles of 4, 2, 1 copies of the original item, and decide sequentially upon putting these subitems in the knapsack. Figure 1 shows that the 3 binary sub-decisions are not pairwise independent: The decision tree has 3 levels, corresponding to the 3 subitems. We denote a decision to put (not to put) an item in the knapsack by "+" ("-"), respectively. The figure shows that if we decide to put the item of weight 4 (leftmost uppermost fork), we cannot put the subitem of weight 2, so the constraint $u_1 = 5$ becomes binding. On the other hand, if we decide not to put the subitem of weight 4 in the knapsack (rightmost uppermost fork), the remaining 2 sub-decisions are independent of each other.

In the DP formulations (2) and (7) we encode whether the constraint is or is not binding in the third subscript of the variable (denoted by "r").

3 Algorithm via a primal DP formulation

A pseudo-polynomial algorithm is achieved using the following recurrence:

$$\begin{array}{lll} s_i(j) & = \sum_{k=0}^{m_i(j)} s_{i-1}(j-kw_i) & 2 \le i \le n,\ j=1,\dots,C, \\ s_1(j) & = m_1(j)+1 & j=1,\dots,C, \end{array} \tag{1}$$

where function $m_i : [0,\dots,C] \to \mathbb{Z}^+$ is defined as $m_i(j) := \max\{x \in \mathbb{Z}^+ \mid x \le u_i,\ xw_i \le j\}$ and returns the maximum number of copies of item i that can be placed in a knapsack with capacity j. Here $s_i(j)$ is the number of integer knapsack solutions that use a subset of the items $\{1,\dots,i\}$ whose weights sum up to at most j. The solution of the counting problem is therefore $s_n(C)$. The complexity of this pseudo-polynomial algorithm is $O(nUC)$, i.e., exponential in both the (binary) sizes of U and C. We call such formulation *primal* because the range of the functions in (1) is the number of solutions.

In order to get our FPTAS we give in Section 3.2 a more careful DP formulation which is exponential only in the (binary) size of C. Before doing so, we briefly overview the technique of K-approximation sets and functions in Section 3.1. We use this technique in order to get our first FPTAS. In Section 2 we introduce the idea behind the notion of binding constraints. We use this notion in order to get both FPTASs.

3.1 K-approximation sets and functions

Halman *et al.* [5] have introduced the technique of K-approximation sets and functions, and used it to develop an FPTAS for a certain stochastic inventory control problem. Halman *et al.* [4] have applied this tool to develop a framework for constructing FPTASs for a rather general class of stochastic DPs. This technique has been used to yield FPTASs to various optimization problems, see [4] and the references therein. In this section we provide an overview of the technique of K-approximation sets and functions. In the next section we use this tool to construct FPTASs for counting the number of solutions of the integer knapsack problem. To simplify the discussion, we modify Halman *et al.*'s definition of the K-approximation function by restricting it to integer-valued nondecreasing functions.

Let $K \ge 1$, and let $\varphi : \{0,\dots,B\} \to Z^+$ be an arbitrary function. We say that $\tilde{\varphi} : \{0,\dots,B\} \to Z^+$ is a *K-approximation function* of φ if $\varphi(x) \le \tilde{\varphi}(x) \le K\varphi(x)$ for all $x = 0,\dots,B$. The following property of K-approximation functions is extracted from Proposition 5.1 of [4], which provides a set of general computational rules of K-approximation functions. Its validity follows directly from the definition of K-approximation functions.

▶ **Property 3.** *For $i = 1,2$ let $K_i \ge 1$, let $\varphi_i : \{0,\dots,B\} \to Z^+$ and let $\tilde{\varphi}_i : \{0,\dots,B\} \to Z^+$ be a K_i-approximation of φ_i. The following properties hold:*

Summation of approximation: *$\tilde{\varphi}_1+\tilde{\varphi}_2$ is a $\max\{K_1,K_2\}$-approximation function of $\varphi_1+\varphi_2$.*

Approximation of approximation: *If $\varphi_2 = \tilde{\varphi}_1$ then $\tilde{\varphi}_2$ is a K_1K_2-approximation function of φ_1.*

Let $K > 1$. Let $\varphi : \{0,\dots,B\} \to Z^+$ be a nondecreasing function and $W = \{k_1,k_2,\dots,k_r\}$ be a subset of $\{0,\dots,B\}$, where $0 = k_1 < k_2 < \cdots < k_r = B$. We say that W is a *K-approximation set* of φ if $\varphi(k_{j+1}) \le K\varphi(k_j)$ for each $j = 1,2,\dots,r-1$ that satisfies $k_{j+1} - k_j > 1$. This means that the *values* of φ on consecutive points of the approximation set essentially form a geometric progression with ratio of approximately K. (Consecutive points of the approximation set *itself* do not necessarily form a geometric sequence.) It is easy to see that given φ, there exists a K-approximation set of φ with cardinality $O(\log_K M)$,

Algorithm 1 FUNCTION COMPRESS(φ, K) returns a step nondecreasing K-approximation of φ

```
1: Function Compress($\varphi, K$)
2: obtain a $K$-approximation set $W$ of $\varphi$
3: return the $K$-approximation function of $\varphi$ induced by $W$
```

where M is any constant upper bound of $\varphi(\cdot)$. Furthermore, this set can be constructed in $O\big((1+t_\varphi)\log_K M \log_2 B\big)$ time, where t_φ is the amount of time required to evaluate φ (see [4, Prop. 4.6] for a formal proof).

Given φ and a K-approximation set $W = \{k_1, k_2, \ldots, k_r\}$ of φ, a K-approximation function of φ can be obtained easily as follows [4, Def.4.4]: Define $\hat{\varphi} : \{0, \ldots, B\} \to Z^+$ such that

$$\hat{\varphi}(x) = \varphi(k_j) \qquad k_{j-1} < x \le k_j \text{ and } j = 2, \ldots, r,$$

and that

$$\hat{\varphi}(k_1) = \varphi(k_1).$$

Note that $\varphi(x) \le \hat{\varphi}(x) \le K\varphi(x)$ for $x = 0, \ldots, B$. Therefore, $\hat{\varphi}$ is a nondecreasing K-approximation function of φ. We say that $\hat{\varphi}$ is the *K-approximation function of φ induced by W*.

The procedure for the construction of a K-approximation function $\tilde{\varphi}$ for φ is stated as Algorithm 1[1]. By applying approximation of approximation in Property 3 and the discussion above we get the following result (see also [4, Prop. 4.5]).

▶ **Proposition 4.** *Let $K_1, K_2 \ge 1$ be real numbers, $M > 1$ be an integer, and let $\varphi : [0, \ldots, B] \to [0, \ldots, M]$ be a nondecreasing function. Let $\bar{\varphi}$ be a nondecreasing K_2-approximation function of φ. Then Function* COMPRESS*($\bar{\varphi}, K_1$) returns in $O((1+t_{\bar{\varphi}})(\log_K M \log B))$ time a nondecreasing step function $\tilde{\varphi}$ with $O(\log_{K_1} M)$ steps which K_1K_2-approximates φ. The query time of $\tilde{\varphi}$ is $O(\log\log_{K_1} M)$ if it is stored in a sorted array $\{(x, \tilde{\varphi}) \mid x \in W\}$.*

Halman *et al.* have designed a framework that yields FPTASs for DPs that posses a certain monotone structure [4]. (We say that a DP has *depth T* if it consists of T sequential recursive equations. In our case the state space of the DP consists of all possible remaining capacities in the knapsack and the action space is binary – to put or not to put a certain (sub)item in the knapsack.)

▶ **Theorem 5.** *(Adapted from [4, Thm. 8.2]) A monotone DP with depth T, $|$action space$| \le A$, $|$state space$| \le S$ and bound M on the maximal value of the solution admits an $O(\frac{T^2}{\epsilon} A \log S \log M \log \frac{T \log M}{\epsilon})$ time FPTAS.*

We note in passing that the original result [4, Thm. 8.2] is stated for very large (exponential) action spaces. Theorem 5 is a version modified for action spaces of size polynomial in the input size and is achieved by performing the minimization in [4, equation (7.1)] over the entire action space. See the proof of [4, Thm. 8.2] for detail about algorithm analysis.

[1] The author thanks Jim Orlin for suggesting the presentation of this function, as well as the term "Compress".

3.2 A more efficient DP formulation

In this section we reformulate (1) as a DP that can be solved in time pseudo-polynomial in the (binary) size of C only. As explained above, instead of deciding at once how many copies of item i to put in the knapsack, we break the decision into $\lfloor \log^+ m_i(j) \rfloor + 1$ binary sub-decisions. Sub-decision $\ell = 1, \dots, \lfloor \log^+ m_i(j) \rfloor + 1$ checks the possibility of putting $2^{\ell-1}$ copies of item i in the knapsack. We do so using, what we call, the idea of *binding constraints*. For $\ell \geq 1$ let $z_{i,\ell,0}(j)$ be the number of solutions for a knapsack of capacity j that use a subset of the items $\{1, \dots, i\}$, put no more than $2^\ell - 1$ copies of item i, and no more than u_k copies of item k, for $k = 1, \dots, i-1$. For $\ell \geq 1$ let $z_{i,\ell,1}(j)$ be the number of solutions for a knapsack of capacity j that use a subset of the items $\{1, \dots, i\}$, put no more than u_i mod 2^ℓ copies of item i, and no more than u_k copies of item k, for $k = 1, \dots, i-1$. In this way, considering the third index of $z_{i,\ell,r}(j)$, if $r = 0$ then the constraint $x \leq u_i$ is assumed to be *non binding*. If, on the other hand, $r = 1$ then the constraint $x \leq u_i$ may be *binding*. Before giving the formal recurrences we need a few definitions. Let $\log^+ x := \max\{0, \log x\}$. Let $\mathrm{msb}(x, i) := \lfloor \log(x \mod 2^i) \rfloor + 1$. $\mathrm{msb}(x, i)$ is therefore the most significant 1-digit of $(x \mod 2^i)$ if $(x \mod 2^i) > 0$, and is $-\infty$ otherwise. E.g., $\mathrm{msb}(5, 2) = 1$ and $\mathrm{msb}(4, 1) = -\infty$. Our recurrences are as follows:

$$z_{i,\ell,0}(j) = z_{i,\ell-1,0}(j) + z_{i,\ell-1,0}(j - 2^{\ell-1} w_i) \qquad \ell = 2, \dots, \lfloor \log^+ m_i(j) \rfloor + 1\,, \tag{2a}$$

$$z_{i,\ell,1}(j) = z_{i,\ell-1,0}(j) + z_{i,\mathrm{msb}(u_i,\ell-1),1}(j - 2^{\ell-1} w_i) \qquad \ell = 2, \dots, \lfloor \log^+ m_i(j) \rfloor + 1\,, \tag{2b}$$

$$z_{i,1,r}(j) = \begin{array}{l} z_{i-1,\lfloor \log^+ m_{i-1}(j) \rfloor + 1,1}(j) + \\ + z_{i-1,\lfloor \log^+ m_{i-1}(j - w_i) \rfloor + 1,1}(j - w_i)\,, \end{array} \tag{2c}$$

$$z_{i,-\infty,1}(j) = z_{i-1,\lfloor \log^+ m_{i-1}(j) \rfloor + 1,1}(j)\,, \tag{2d}$$

$$z_{1,\ell,r}(j) = m_1(j) + 1 \qquad \ell = 1, \dots, \lfloor \log^+ m_1(j) \rfloor + 1\,, \tag{2e}$$

$$z_{i,\ell,r}(j) = 0 \qquad j < 0, \tag{2f}$$

where $r = 0, 1\,,\ i = 2, \dots, n\,,$ and $j = 0, \dots, C\,,$ unless otherwise specified. The solution of the counting problem is therefore $z_{n,\lfloor \log u_n \rfloor + 1,1}(C)$. The time needed to solve this program is only $O(nC \log U)$.

We now explain the six equations in formulation (2) in more detail. Equation (2a) deals with the case where the constraint $x \leq u_i$ is non binding, so putting $2^\ell - 1$ more copies of item i in a knapsack of remaining capacity j is a feasible possibility. Clearly, in the following steps the constraint $x \leq u_i$ remains non binding. As for equation (2b), it deals with the case where the constraint $x \leq u_i$ may be binding when putting $2^{\ell-1}$ copies of item i in the knapsack. If we do put this number of copies, the constraint may be binding, otherwise it is assured to be non binding. Equation (2c) deals with the possibility of putting an odd number of copies of item i in the knapsack. Equation (2d) is only called by equation (2b), when *exactly* u_i copies of item i are put in the knapsack. Equation (2e) deals with the initial condition of one element only, and the last equation deals with the boundary condition that there is not enough capacity in the knapsack.

In order to design an FPTAS to our problem, we first extend the DP formulation (2) to any integer positive index ℓ by letting $z_{i,\ell,r}(j) = 0$ for $i = 1, \dots, n\,, r = 0, 1$ and $\ell > \lfloor \log^+ m_i(j) \rfloor + 1$. (Note that without this extension $z_{i,\ell,r}(\cdot)$ is not necessarily defined over the entire interval $(-\infty, \dots, C]$. Moreover, this extended formulation assures that $z_{i,\ell,r}(\cdot)$ is monotone nondecreasing.) We denote this extended set of recurrences by (3). The solution of the counting problem via (3) remains $z_{n,\lfloor \log u_n \rfloor + 1,1}(C)$.

From the fact that $z_{i,\ell,r}(\cdot)$ are monotone nondecreasing functions and that the action is binary, one can apply Theorem 5 with parameters set to $T \sim O(n \log U), A = 2, S = C$ and

Algorithm 2 FPTAS for counting integer knapsack.

```
1: Function CountIntegerKnapsackPrimal$(w, C, u, \epsilon)$
2: $K \leftarrow \sqrt[(n-1)(\lfloor \log U \rfloor+1)+1]{1+\epsilon}$
3: for $\ell := 1$ to $\lfloor \log u_1 \rfloor + 1$ and $r = 0, 1$ do $\tilde{z}_{1,\ell,r} \leftarrow$ Compress$(z_{1,\ell,r}, K)$ /* $z_{1,\ell,r}$ as defined in (3) */
4: for $i := 2$ to $n$ do
5:     $\tilde{z}_{i,-\infty,1}(\cdot) \leftarrow \tilde{z}_{i-1,\lfloor \log^+ m_{i-1}(\cdot) \rfloor+1,1}(\cdot)$
6:     for $r = 0, 1$ do $\tilde{z}_{i,1,r}(\cdot) \leftarrow$ Compress$(\tilde{z}_{i-1,\lfloor \log^+ m_{i-1}(\cdot) \rfloor+1,1}(\cdot) + \tilde{z}_{i-1,\lfloor \log^+ m_{i-1}(\cdot - w_i) \rfloor+1,1}(\cdot - w_i), K)$
7:     for $\ell := 2$ to $\lfloor \log u_i \rfloor + 1$ do
8:         $\tilde{z}_{i,\ell,0}(\cdot) \leftarrow$ Compress$(\tilde{z}_{i,\ell-1,0}(\cdot) + \tilde{z}_{i,\ell-1,0}(\cdot - 2^{\ell-1} w_i), K)$
9:         $\tilde{z}_{i,\ell,1}(\cdot) \leftarrow$ Compress$(\tilde{z}_{i,\ell-1,0}(\cdot) + \tilde{z}_{i,\mathrm{msb}(u_i,\ell-1),1}(\cdot - 2^{\ell-1} w_i), K)$
10:    end for
11: end for
12: return $\tilde{z}_{n,\lfloor \log u_n \rfloor+1,1}(C)$
```

$M = U^n$ and get an $O(\frac{n^3}{\epsilon} \log^3 U \log C \log \frac{n \log U}{\epsilon})$ time FPTAS. For the sake of completeness, in the next section we explicitly state the FPTAS for our problem and sketch its analysis.

3.3 Algorithm statement

The idea behind our approximation algorithm is to compute an approximation for $z_{n,\lfloor \log u_n \rfloor+1,1}(C)$ by using the recurrences in (3). This is done by recursively computing K-approximation functions for the $O(\sum_{i=1}^n \lfloor \log u_i \rfloor)$ different functions in (3). Due to summation of approximation coupled with approximation of approximation (Property 3) there is a deterioration of at most factor K between the ratio of approximation of $z_{i,\ell,r}$ and that of $z_{i,\ell-1,r}$ (for $\ell > 1$), as well as between the ratio of approximation of $z_{i,1,r}$ and that of $z_{i-1,\lfloor \log u_{i-1} \rfloor+1,r}$. Therefore, by choosing $K = \sqrt[(n-1)(\lfloor \log U \rfloor+1)+1]{1+\epsilon}$ one gets that the total accumulated multiplicative error over the entire algorithm does not exceed $1+\epsilon$. For a given instance (w, C, u) of the integer knapsack problem and a tolerance parameter $\epsilon \in (0, 1]$, our approximation algorithm is formally given as Function CountIntegerKnapsackPrimal(w, C, u, ϵ), see Algorithm 2. (From hereon after we use the notation $z(\cdot)$, where the "$\cdot$" stands for the argument of function z. E.g., the value of $z(\cdot - w)$ for variable value 2 is $z(2-w)$. Put it differently, the function z is shifted by $-w$.)

The proof that CountIntegerKnapsackPrimal(w, C, u, ϵ) returns an approximated number of solutions that varies from the exact number of solutions by relative error of at most ϵ is done by double induction over i and ℓ, and shows that $\tilde{z}_{i,\ell,r}$ is a $K^{(i-2)(\lfloor \log U \rfloor+1)+\ell+1}$-approximation of $z_{i,\ell,r}$ for $i = 2, \ldots, n$, $\ell = 0, \ldots, \lfloor \log u_i \rfloor + 1$ and $r = 0, 1$. See the full version of this paper for a formal proof.

We next analyze the complexity of the algorithm. Clearly, the running time of the algorithm is dominated by the operations done in the inner for-loop, i.e., steps 8-9, which are executed $O(n \log U)$ times. We analyze, w.l.o.g., a single execution of step 8. By Proposition 4, the query time of each of the $\tilde{z}_{i,\ell-1,0}(\cdot)$ and $\tilde{z}_{i,\ell-1,0}(\cdot)$ is $O(\log \log_K M)$, where M is an upper bound on the counting problem, e.g., $M = U^n$. Therefore, applying again Proposition 4, each call to Compress runs in $O(\log_K M \log C \log \log_K M)$ time. Using the inequality $(1+\frac{x}{n})^n \le 1+2x$ which holds for $0 \le x \le 1$ we get that $K \ge 1 + \frac{\epsilon}{2\left((n-1)(\lfloor \log U \rfloor+1)+1\right)}$. Using the inequality $\log(1+y) \ge y$ which holds for $y \in [0, 1]$, and changing the bases of the logarithms to two, we get that the overall running time of the algorithm is $O(\frac{n^3}{\epsilon} \log^3 U \log C \log \frac{n \log U}{\epsilon})$.

4 Algorithm via a dual DP formulation

In this section we provide an FPTAS to counting integer knapsack solutions using the analysis of [10] for counting 0/1 knapsack solutions. Our FPTAS will be faster than the one presented in the previous section by a factor of $\log U \log C$.

4.1 The 0/1 knapsack

In this section we present the main ideas used to derive the FPTAS to counting 0/1 knapsack solutions [10, Sec. 2]. Štefankovič *et al.* [10] begin by defining a dual DP formulation as follows. For $i = 1, \dots, n$ let $\tau_i(a)$ be the smallest capacity C such that there exist at least a solutions to the knapsack problem with items $1, 2, \dots, i$ and capacity C. Using standard conventions, the value of τ_0 is given by

$$\tau_0(a) = \begin{cases} -\infty & \text{if } a = 0\,, \\ 0 & \text{if } 0 < a \le 1\,, \\ \infty & \text{otherwise.} \end{cases} \tag{4}$$

It follows that the number of knapsack solutions satisfies $Z = \max\{a \mid \tau_n(a) \le C\}$. [10, Lem. 2.1] states that $\tau_i(a)$ satisfies the following recurrence:

$$\tau_i(a) = \min_{\alpha \in [0,1]} \max \begin{cases} \tau_{i-1}(\alpha a), \\ \tau_{i-1}((1-\alpha)a) + w_i\,. \end{cases} \tag{5}$$

Intuitively, to obtain a solutions that consider the first i items, we need to have, for some $\alpha \in [0, 1]$, αa solutions that consider the first $i - 1$ items and $(1 - \alpha)a$ solutions that contain the ith item and consider the first $i - 1$ items. The recursion tries all possible values of α and take the one that yields the smallest (optimal) value for $\tau_i(a)$. We call such formulation *dual* because the range of the functions in (4)-(5) is the capacity of the knapsack.

[10] then move to an approximation of τ that can be computed efficiently and define function $T : \{0, \dots, s\} \to \mathbb{R}^+ \cup \{\infty\}$ which only considers a small subset of values a for the argument in $\tau(\cdot)$, these values form a geometric progression. Let

$$T_0(a) = \begin{cases} -\infty & \text{if } a = 0\,, \\ 0 & \text{if } 0 < a \le 1\,, \\ \infty & \text{otherwise,} \end{cases}$$

and let

$$Q := 1 + \frac{\epsilon}{n+1}, \qquad s := \lceil \log_Q 2^n \rceil = O(n^2/\epsilon).$$

The functions $T_i(\cdot)$ are defined via the recurrence (5) that the function τ satisfies. Namely, T is defined by the following recurrence:

$$T_i(j) = \min_{\alpha \in [0,1]} \max \begin{cases} T_{i-1}(j + \log_Q \alpha), \\ T_{i-1}(j + \log_Q (1-\alpha)) + w_i\,. \end{cases} \tag{6}$$

The FPTAS computes all $T_i(\cdot)$ exhaustively and returns $Q^{j'+1}$, where $j' := \max\{j \mid T_n(j)) \le C\}$, see [10] for the analysis of the FPTAS.

4.2 The dual DP formulation

In what follows we show that even if the values of the u_i are not all powers of 2 (minus one) we can still give a recurrence using, what we call, the idea of *binding constraints*. For $\ell \geq 1$ let $\tau_{i,\ell,0}(a)$ be the minimal knapsack capacity needed so that there are at least a solutions that use a subset of the items $\{1, \ldots, i\}$, put no more than $2^\ell - 1$ copies of item i, and no more than u_k copies of item k, for $k = 1, \ldots, i-1$. For $\ell \geq 1$ let $\tau_{i,\ell,1}(a)$ be the minimal knapsack capacity needed so that there are at least a solutions that use a subset of the items $\{1, \ldots, i\}$, put no more than $u_i \mod 2^\ell$ copies of item i, and no more than u_k copies of item k, for $k = 1, \ldots, i-1$. In this way, considering the third index of $\tau_{i,\ell,r}(a)$, if $r = 0$ then the constraint $x \leq u_i$ is assumed to be *non binding*. If, on the other hand, $r = 1$ then the constraint $x \leq u_i$ may be *binding*. Our recurrences are as follows (for simplicity we set $u_0 = 1$. Recall that the definition of $\mathrm{msb}(\cdot)$ is given in Section 3.2):

$$\tau_{i,\ell,0}(a) = \min_{\alpha \in [0,1]} \max \begin{cases} \tau_{i,\ell-1,0}(\alpha a), \\ \tau_{i,\ell-1,0}((1-\alpha)a) + 2^{\ell-1} w_i \end{cases} \qquad \ell = 2, \ldots, \lfloor \log u_i \rfloor + 1 \tag{7a}$$

$$\tau_{i,\ell,1}(a) = \min_{\alpha \in [0,1]} \max \begin{cases} \tau_{i,\ell-1,0}(\alpha a), \\ \tau_{i,\mathrm{msb}(u_i,\ell-1),1}((1-\alpha)a) + 2^{\ell-1} w_i \end{cases} \qquad \ell = 2, \ldots, \lfloor \log u_i \rfloor + 1 \tag{7b}$$

$$\tau_{i,1,r}(a) = \min_{\alpha \in [0,1]} \max \begin{cases} \tau_{i-1,\lfloor \log u_{i-1} \rfloor + 1,1}(\alpha a), \\ \tau_{i-1,\lfloor \log u_{i-1} \rfloor + 1,1}((1-\alpha)a) + w_i \end{cases} \qquad r = 0, 1 \tag{7c}$$

$$\tau_{i,-\infty,1}(a) = \tau_{i-1,\lfloor \log u_{i-1} \rfloor + 1,1}(a) \tag{7d}$$

$$\tau_{0,1,1}(a) = \begin{cases} -\infty & \text{if } a = 0\,, \\ 0 & \text{if } 0 < a \leq 1\,, \\ \infty & \text{otherwise.} \end{cases} \tag{7e}$$

where $i = 1, \ldots, n$. The number of knapsack solutions satisfies

$$Z = \max\{a \mid \tau_{n,\lfloor \log^+ u_n \rfloor + 1,1}(a) \leq C\}.$$

We now explain the five equations in formulation (7) in more detail. Equation (7a) deals with the case where the constraint $x \leq u_i$ is non binding, so placing in the knapsack $2^\ell - 1$ more copies of item i is a feasible possibility. Clearly, in the following steps the constraint $x \leq u_i$ remains non binding. As for equation (7b), it deals with the case where the constraint $x \leq u_i$ may be binding when putting $2^{\ell-1}$ copies of item i in the knapsack. If we do put this number of copies, the constraint may be binding and at most $u_i \mod 2^{\ell-1}$ more copies can be placed in the knapsack. Otherwise it is assured to be non binding. Equation (7c) deals with the possibility of placing in the knapsack an odd number of copies of item i. As for equation (7d), note that it is called by equation (7b) when *exactly* u_i copies of item i are put in the knapsack. Equation (7e) is a boundary condition similar to (4).

We now define an approximation $T_{i,\ell,r}$ of $\tau_{i,\ell,r}$ similarly to the 0/1-knapsack case, but where

$$Q := 1 + \frac{\epsilon}{(n+1)\log U}, \qquad s := \lceil \log_Q(U^n) \rceil = O\left(\frac{n^2 \log^2 U}{\epsilon}\right).$$

The function $T_{i,\ell,r}$ is defined using the recurrence (7). E.g., using (7a) we define:

$$T_{i,\ell,0}(j) = \min_{\alpha \in [0,1]} \max \begin{cases} T_{i,\ell-1,0}(j + \log_Q \alpha), \\ T_{i,\ell-1,0}(j + \log_Q(1-\alpha)) + 2^{\ell-1} w_i\,. \end{cases} \tag{8}$$

Algorithm 3 FPTAS for counting integer knapsack via dual DP formulation.

```
1: Function CountIntegerKnapsackDual($w, C, u, \epsilon$)
2: $Q \leftarrow 1 + \frac{\epsilon}{(n+1)\log U}$, $s \leftarrow \lceil \log_Q(U^n) \rceil$, $T_{0,1,1}(0) \leftarrow 0$, $T_{0,1,1}(j) \leftarrow \infty$ for $j > 0$
3: for $i := 1$ to $n$ do
4:     By convention, $T_{i,\ell,r}(k) \leftarrow 0$ for $\ell \geq 1, r = 0, 1$ and $k < 0$
5:     Calculate $T_{i,-\infty,1}(\cdot)$ via the analogue of equation (7d)
6:     for $r = 0, 1$ do Calculate $T_{i,1,r}(\cdot)$ via the analogue of equation (7c)
7:     for $\ell := 2$ to $\lfloor \log u_i \rfloor + 1$ do
8:         Calculate $T_{i,\ell,0}(\cdot)$ via the analogue of equation (7a), i.e., via equation (8)
9:         Calculate $T_{i,\ell,1}(\cdot)$ via the analogue of equation (7b)
10:    end for
11: end for
12: $j' \leftarrow \max\{j \mid T_{n,\lfloor \log u_n v \rfloor,1}(j) \leq C\}$
13: return $Q^{j'+1}$
```

4.3 Algorithm statement

Similarly to the algorithm given in [10], also our algorithm computes all $T_{i,\ell,r}(\cdot)$ exhaustively and returns $Q^{j'+1}$, where $j' := \max\{j \mid T_{n,\lfloor \log u_n v \rfloor,1}(j)) \leq C\}$. For a given instance (w, C, u) of the integer knapsack problem and a tolerance parameter $\epsilon \in (0, 1]$, our approximation algorithm is stated as Algorithm 3.

We now outline an analysis of the running time of Algorithm 3. Since the arguments of function $T_{i,\ell,r}$ in (8) and alike are step functions of α, it suffices to consider a discrete set of α which yields all possible values of the arguments. Such set is of cardinality $O(s)$. Because the various $T_{i,\ell,r}$ are nondecreasing functions of α, the minima in (8) and alike can be computed in time $O(\log s)$ via binary search. Note that there are $O(ns)$ entries of the various functions. As explained in the analysis in [10], the algorithm can be implemented in $O(ns \log s)$ time. Since we have $s = O(\frac{n^2 \log^2 U}{\epsilon})$, the algorithm can be implemented in $O(\frac{n^3 \log^2 U}{\epsilon} \log \frac{n \log U}{\epsilon})$ time, as indicated in Theorem 1. We note that the running time of the algorithm differs from the one of [10] because in the latter case we have a different value of s, i.e., $s = O(\frac{n^2}{\epsilon})$.

5 Concluding remarks

In this paper we present two deterministic FPTASs for counting integer knapsack solutions, each of which improves upon the best known results. Both FPTASs relay on clever DP formulations and the new DP technique of binding constraints. The only strongly polynomial approximation scheme for this problem is a (randomized) FPRAS [1]. It is an open problem to design an FPTAS that is both strongly-polynomial and deterministic. It is also an open problem to design an FPTAS for the multidimensional knapsack problem that is more efficient than the one of [2].

References

1 M. E. Dyer. Approximate counting by dynamic programming. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC), June 9-11, 2003, San Diego, CA, USA*, pages 693–699, 2003.

2 P. Gopalan, A. Klivans, and R. Meka. Polynomial-time approximation schemes for Knapsack and related counting problems using branching programs. *CoRR*, abs/1008.3187, 2010.

3 P. Gopalan, A. Klivans, R. Meka, D. Štefankovič, S. Vempala, and E. Vigoda. An FPTAS for #Knapsack and related counting problems. In *IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 817–826, 2011.

4 N. Halman, D. Klabjan, C.-L. Li, J. Orlin, and D. Simchi-Levi. Fully polynomial time approximation schemes for stochastic dynamic programs. *SIAM Journal on Discrete Mathematics*, 28:1725–1796, 2014.

5 N. Halman, D. Klabjan, M. Mostagir, J. Orlin, and D. Simchi-Levi. A fully polynomial time approximation scheme for single-item stochastic inventory control with discrete demand. *Mathematics of Operations Research*, 34:674–685, 2009.

6 M. Jerrum and A. Sinclair. The Markov chain Monte Carlo method: An approach to approximate counting and integration. In D.S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 482–520. PWS Publishing Company, Boston, 1996.

7 N. Megiddo. On the complexity of linear programming. In *Advances in economic theory*, pages 225–268, Cambridge, UK, 1989. Econom. Soc. Monogr. 12.

8 R. Meka and D. Zuckerman. Pseudorandom generators for polynomial threshold functions. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 427–436, 2010.

9 R. Rizzi and A. Tomescu. Faster FPTASes for counting and random generation of knapsack solutions. In *Proceedings of the 22nd Annual European Symposium on Algorithms (ESA)*, pages 762–773, 2014.

10 D. Štefankovič, S. Vempala, and E. Vigoda. A deterministic polynomial-time approximation scheme for counting knapsack solutions. *SIAM Journal on Computing*, 41:356–366, 2012.

A Competitive Flow Time Algorithm for Heterogeneous Clusters Under Polytope Constraints

Sungjin Im[*1], **Janardhan Kulkarni**[2], **Benjamin Moseley**[†3], **and Kamesh Munagala**[‡4]

1 **EECS, University of California at Merced, Merced, CA, USA**
sim3@ucmerced.edu

2 **Microsoft Research, Redmond, WA, USA**
jakul@microsoft.com

3 **Department of Computer Science and Engineering, Washington University, St. Louis, MO, USA**
bmoseley@wustl.edu

4 **Department of Computer Science, Duke University, Durham, NC, USA**
kamesh@cs.duke.edu

Abstract

Modern data centers consist of a large number of heterogeneous resources such as CPU, memory, network bandwidth, etc. The resources are pooled into clusters for various reasons such as scalability, resource consolidation, and privacy. Clusters are often heterogeneous so that they can better serve jobs with different characteristics submitted from clients. Each job benefits differently depending on how much resource is allocated to the job, which in turn translates to how quickly the job gets completed.

In this paper, we formulate this setting, which we term MULTI-CLUSTER POLYTOPE SCHEDULING (MCPS). In MCPS, a set of n jobs arrive over time to be executed on m *clusters*. Each cluster i is associated with a polytope $\mathcal{P}_i$, which constrains how fast one can process jobs assigned to the cluster. For MCPS, we seek to optimize the popular objective of minimizing average weighted flow time of jobs in the online setting. We give a constant competitive algorithm with small constant resource augmentation for a large class of polytopes, which capture many interesting problems that arise in practice. Further, our algorithm is non-clairvoyant. Our algorithm and analysis combine and generalize techniques developed in the recent results for the classical unrelated machines scheduling and the polytope scheduling problem [10, 12, 11].

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problem]: Sequencing and scheduling

Keywords and phrases Polytope constraints, average flow time, multi-clusters, online scheduling, and competitive analysis.

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.10

[*] S. Im was supported in part by NSF grants CCF-1409130 and CCF-1617653.
[†] B. Moseley was supported in part by NSF grant CCF-1617724, a Yahoo faculty award and a Google faculty award.
[‡] K. Munagala was supported by NSF grants CCF-1408784, IIS- 1447554, and CCF-1348696.

1 Introduction

Modern data centers consist of a large number of machines, each with its own resources such as CPU, memory, network etc, that are organized into hundreds of clusters. Typically, jobs are data intensive and require a lot of resources for running. Examples of such jobs can be found in MapReduce systems. At such large scales, the resources are assumed to be continuously divisible, thus can be compactly represented as a vector consisting of CPU, memory, network bandwidth, etc. Efficiently partitioning these resources among jobs is a major challenge in system design. Complicating the scheduling decisions further is the fact that jobs have different characteristics, and may get different benefits/utilities even when assigned the same resources – some jobs are CPU intensive while others require more memory. In fact, this *multi-dimensional* nature is a key factor that differentiates data center scheduling from other well-studied scheduling settings. Due to the explosive growth of data centers and the associated operating costs, the multi-dimensional scheduling problems have gained a lot of attention in the systems literature recently; see [8] and follow-up work [4, 18, 9, 1, 2, 17, 15].

Various scheduling problems in the presence of multiple resources can be modeled by a polytope which constrains the rate at which jobs can be processed. This abstraction of the data center scheduling problems was introduced under the name Polytope Scheduling Problem (PSP) in a recent work by Im et al. [10, 11].

Unfortunately, despite its generality, PSP model fails to accurately capture the system architectures where machines are grouped into clusters. By grouping machines into clusters, the system can restrict the set of jobs that share a given cluster. Moreover, clusters in data centers are often heterogeneous in that some clusters are more suitable for certain types of jobs than others. For example, jobs may get processed more effectively on some clusters due to their proximity to data. Further, the location of data can also change the resource requirements of jobs – if a job is assigned to the cluster on which its data is located then it does not need access to the network when executing. Data centers also have special purpose hardware (such as FPGA/GPUs) for faster execution of certain jobs. Thus, the processing times of jobs can also depend on the clusters that they are assigned to.

In this paper, we introduce a more realistic scheduling setting where each cluster is associated with a distinct polytope that determines the rates at which jobs get processed. At a high level, this not only captures the *multi-dimensional* nature found in PSP, but also captures the *unrelated* aspect reminiscent of the classical scheduling literature [3]. In the classical unrelated machines model, each job can get processed at a completely different rate depending on its machine assignment. Thus, our model lifts the unrelated nature from machines to clusters while staying faithful to the multi-dimensional aspect of data center scheduling problems.

We focus on minimizing the average (weighted) flow time of jobs. A job's flow time measures how long the job waits from its arrival until its completion, thus the average (or equivalently total) flow time measures the average delay experienced by clients. For this popular objective, competitive algorithms are known for some special cases of PSP and also for the unrelated machines setting. However, their analyses use two very different methods – dual fitting for the unrelated machines setting [12], and potential function argument for PSP [11]. In this paper, we combine the two different algorithms used in [12, 11] and develop a new potential function analysis that *unifies* the two disparate analyses from the previous works.

1.1 Problem Definition

The Multi-Cluster Polytope Scheduling (MCPS), which is a generalization of the Polytope Scheduling Problem (PSP) [10], is defined as follows. A set of n jobs arrive over time. Each job j has a weight w_j, size (or processing length) p_j, and arrival time r_j, and needs to be processed using a set of m clusters. Each cluster i is associated with a convex polytope $\mathcal{P}_i$ that constrains the feasible space of rates of jobs assigned to the cluster. Each polytope $\mathcal{P}_i$ is defined over the entire set of jobs and is assumed to be downward-closed, meaning that if $\vec{y} \in \mathcal{P}_i$ and $\vec{z} \leq \vec{y}$, then any $\vec{z} \in \mathcal{P}_i$. The scheduler has to assign rates to jobs, $\{y_j\}_j$, subject to the polytope constraints and a natural requirement that a job can be scheduled only on a single cluster at any given instant of time. Applications of MCPS will be discussed in Section 1.2.

Note that we allow a job to be processed on more than one cluster over the course of its execution. However, at any given time a job can be processed on only one cluster. This is exactly the PSP problem [10] when there is only one cluster.

In this paper we seek to design *online* scheduling algorithms for MCPS, which have to make scheduling decisions only based on the jobs that have arrived. In other words, the online scheduler learns about a job j along with its properties when it arrives. Our goal is to minimize the total weighted flow time of jobs in the setting of MCPS. Let $y^{\mathcal{A}}_{jt}$ denote the rate at which job j is processed at time t by a scheduler $\mathcal{A}$. Then, job j's completion time $C^{\mathcal{A}}_j$ under the schedule of $\mathcal{A}$ is defined to be the first time t' such that $\int_{t=r_j}^{t'} y^{\mathcal{A}}_{jt} \mathsf{d}t \geq p_j$. Job j's flow time is the length of time job j waits to be completed since its arrival and is defined as $F^{\mathcal{A}}_j = C^{\mathcal{A}}_j - r_j$. Similarly, job j's weighted flow time is defined as $w_j F^{\mathcal{A}}_j$ factoring in the job's weight. When the algorithm $\mathcal{A}$ and time t are clear from the context, we may drop them from the notation. The goal is to minimize $\sum_j w_j F^{\mathcal{A}}_j$.

We will use the standard notion of *competitive ratio* for analyzing our algorithms. An online algorithm is α-competitive if for every finite input instance, the cost incurred by the algorithm is at most α times the cost of some optimal offline solution to the instance. Unfortunately, the standard competitive analysis turns out to be too pessimistic in analyzing flow time related objectives: there are no online algorithms with bounded competitive ratios even in much simpler single dimensional settings [7]. We therefore appeal to the standard speed augmentation analysis [14], where we assume the online algorithm can perform $c > 1$ allocations per time step, while OPT is restricted to allocate at the rate of 1. Our goal is to design algorithms that achieve constant competitive ratios on the total flow time objective using the smallest possible extra speed c.

1.2 Our Results

Our main result is a non-trivial generalization of the result shown for 'monotone' PSP in [11] to the multi-cluster setting.

Proportional Fairness and Monotone Polytope.

The proportional fairness (PF) algorithm, at each instant of time t, assigns rates $\{y_{jt}\}_j$ to jobs by solving the following convex program over the polytope $\mathcal{P}$.

$$\max \sum_j w_j \log y_{jt} \qquad \text{s.t.} \qquad \vec{y}_t \in \mathcal{P} \tag{1}$$

Note that $\mathcal{P}$ is in the definition of PSP. The PF algorithm generalizes the weighted round robin (WRR) algorithm to the multidimensional case. The study of PF even dates back to

Nash's seminal work [16]. The algorithm PF has a very nice market clearing interpretation – It finds prices for resources so that every resource (with a positive price) is completely sold out when each player (job) j with money w_j buys resources to maximize its utility under the prices. Further, the algorithm PF is known to have many desirable fairness properties such as sharing incentiveness, and envy-freeness [8].

▶ **Definition 1** **([11]).** For a subset of jobs S, let $y_j(S)$ denote the rate allocated by PF to job $j \in S$, as given by the equation (1). The PF allocation is said to be *monotone* if for any S and $j' \notin S$, we have the following condition: For all $j \in S$, $y_j(S) \geq y_j(S \cup \{j'\})$. The class Monotone PSP is the sub-class of PSP for which the PF algorithm leads to monotone allocation.

We call MCPS as Monotone-MCPS when the PF algorithm is monotone for every polytope $\mathcal{P}_i$. When there is only one cluster, Im et al. [11] showed that the PF algorithm is $(e+\epsilon)$-speed $O(1/\epsilon^2)$-competitive for the Monotone PSP case. Our first main result is a generalization of this result to the case with an arbitrary number of clusters.

▶ **Theorem 2.** *For Monotone-*MCPS*, there is a $(e+\epsilon)$-speed, $O\left(1/\epsilon^2\right)$-competitive algorithm for minimizing the total weighted flow-time of jobs. Further, our algorithm is non-clairvoyant as it does not make use of the processing lengths of jobs.*

Monotone-PSP captures many important problems such as flow routing to a single sink, routing multicast trees (video-on-demand) etc. We refer the readers to [11] for more details on applications of Monotone-PSP. For completeness, here we give one important class of problems captured by Monotone-PSP that is very relevant to data center scheduling.

Resource Allocation with Substitutes [11]. Consider the multi-dimensional resource allocation problem that arises in scheduling jobs within a cluster. Formally, there are D divisible resources (or dimensions), numbered $1, 2, \ldots, D$. By scaling we can assume w.l.o.g. that each resource is available in a unit supply. If job j is assigned a non-negative vector of resources $\vec{x} = \{x_1, x_2, \ldots, x_D\}$, then the rate at which the job executes is determined by $y_j = u_j(\vec{x})$, where u_j is a concave *utility function* that is known to the scheduler. The constraints $\mathcal{P}$ simply capture that each resource can be allocated to unit amount, i.e. $\sum_j x_{jd} \leq 1$ for all $d \in \{1, 2, \ldots, D\}$. A well-studied special class of utilities in the resource allocation literature are the Constant Elasticity of Scale (CES) utilities, given by:

$$u_j(\vec{x}_j) = \left(\sum_{d=1}^{D} c_{jd} x_{jd}^{\rho_j}\right)^{1/\rho_j}.$$

When $\rho \in (0, 1]$ the utility function captures resources that are *imperfect substitutes* of each other, and the parameter ρ captures the extent of substitutability. A special case as $\rho \to 0$ is termed *Cobb-Douglas* utilities: $u_j(\vec{x}_j) = \prod_{d=1}^{D} x_{jd}^{\alpha_{jd}}$, where $\sum_d \alpha_{jd} \leq 1$ and $\alpha_{jd} \geq 0$ for all j, d. These utilities can be used to model task rates in heterogeneous microprocessor architectures [19]; further, these are widely studied in economics. When $\rho = 1$, CES utilities reduce to *linear utilities.*

It was shown that prove that CES is a special case of Monotone PSP in [11]. Thus, our result immediately gives a competitive algorithm for these problems in the multiple cluster setting.

1.3 High-level Description of the Algorithm

Our algorithm for the Monotone-MCPS consists of two parts: Within a cluster, jobs are assigned rates using the PF algorithm. To decide the assignment of jobs to clusters, we use the Selfish Migrate framework introduced in [12]. In the Selfish Migrate framework, at every time instant, a job behaves like a selfish agent and moves to the cluster that maximizes its own virtual utility function. More precisely, a virtual ordering is maintained among jobs assigned to each cluster i, and a job j's virtual utility is calculated as the speed the job would get on the cluster i under the PF algorithm as if other jobs behind in the ordering were not present. If the job moves to another cluster i', it is placed the last in the virtual ordering of cluster i'. These selfish moves lead to a Nash equilibrium where each job has the best virtual utility on the cluster it is currently assigned to. The monotonicity property of polytopes is crucially used in establishing the existence of such an equilibrium. Thus our algorithm is a *mixture of two interesting equilibria* – a Nash equilibrium across machines under virtual utilities and a market clearing equilibrium on each individual cluster under the algorithm PF. Note that our algorithm does not use job sizes in scheduling decisions, thus is non-clairvoyant.

1.4 Justification for not Encapsulating Clusters into One Polytope

We take a sidestep to clarify some questions that may arise from our definitions of PSP and MCPS. It is fair to ask why one needs to define MCPS when PSP is general enough to model the multiple cluster setting. More precisely, one can define a giant polytope $\mathcal{P}$ that is the union of all polytope constraints $\mathcal{P}_i$, with an extra constraint that at any give time no job can be processed on more than one cluster. Indeed, such a formulation captures MCPS. However, this way of looking at the problem leads to a major technical difficulty: *The giant polytope $\mathcal{P}$ may not inherit the properties satisfied by the individual polytopes $\mathcal{P}_i$.* In particular, the new polytope may not be monotone even if all individual polytopes $\mathcal{P}_i$ are monotone, meaning that we no longer have nice properties that lead to constant-speed constant competitiveness of the PF algorithm. To see this, consider the classical unrelated machines setting. Although it is a special case of PSP, thus our problem, it is not clear if it is a special case of Monotone-PSP. No known techniques can be directly applicable to prove unrelated machines fall into a category of Monotone-PSP. In fact, we conjecture that it is not.

1.5 Our Techniques

Our problem MCPS extends the unrelated machines scheduling and multidimensional scheduling in a natural way. Expectedly, our algorithm for the problem uses a combination of the algorithms developed for the unrelated machines scheduling and the PSP setting. More precisely, within a cluster, our algorithm allocates rates using the Proportional Fairness algorithm similar to the Monotone-PSP case [11] while assigning jobs to clusters using the Selfish Migrate framework as done in the unrelated machine setting [12]. The main technical contribution of this paper lies in unifying the two different analyses of these algorithms – the former uses a potential function argument and the latter a dual-fitting argument.

To unify the two different analyses, one could attempt to use potential function or dual fitting. If one wants to try a dual fitting argument for our problem, the first thing to do would be proving competitiveness of the Monotone-PSP using dual fitting. There are two math programmings involved here: the convex programming (CP) we solve at any instantaneous moment to implement the algorithm PF, and the linear programming (LP) we use to establish the competitiveness of PF for the total weighted flow time objective. One could try to use the values of CP dual variables derived from the KKT conditions of the CP to set the LP dual variables. However, as discussed in [11], the CP dual variables can have highly unstructured values even for monotone-PSP, and this is why [11] used only a CP optimality condition repeatedly, without looking at the dual.

Hence we use a different route by giving an alternative potential function based analysis of the Selfish Migration rule for unrelated machines, and generalizing it to our problem, Monotone-MCPS. Surprisingly, we use an unexpectedly simple potential function, which is just the sum of potential functions for each cluster. The potential for each cluster is defined over the sets of jobs assigned to the cluster by our algorithm and the optimal solution, and there are no terms in the potential connecting different clusters. This is surprising since it has been believed that more sophisticated potential functions should be needed to factor in the changes of the projected objective based on the current assignment of jobs that occur when jobs migrate across different machines.

The reader familiar with potential functions may wonder how we can bound the change of the potential since the jobs the optimal solution assigns to each machine can be vastly different from those our algorithm does. Hence, the credits we get from our algorithm's processing might be completely offset by the debits due to the optimal processing. This is where we use the Selfish Migration rule crucially. At high level, using the fact that each job currently resides on the best machine maximizing its virtual utility, we can safely assume that jobs are assigned following the optimal scheduler since it only gives less credits, which effectively reduces the analysis to each individual machine. However, the extension of this analysis from unrelated machines to Monotone-PSP has another issue. In this thought process of pretending that jobs are assigned to clusters following the optimal solution, we face the challenge of measuring how fast jobs get processed within a cluster under the PF where some of them come from our algorithm's assignment and the other from the optimal solution. We bound such rates using the polytope monotonicity (Proposition 3) and a CP optimality condition (Proposition 4). See Section 2.2 for formal statement of the two properties and how they are used in our analysis.

To summarize, our algorithm, which is a mixture of two equilibria from the Proportional Fairness algorithm and the Selfish Migration rule, is analyzed delicately using the two respective monotonicity properties resulting from the two algorithms: (i) jobs get lower processing rates within a cluster when competing with more jobs; and (ii) each job's migration only increases its virtual utility without hurting other jobs.

Finally, as a byproduct we obtain an alternate analysis of the unrelated machine scheduling to minimize the weighted flow-time in the non-clairvoyant setting. As mentioned before, the classical unrelated machine setting is a special case of the Monotone-MCPS, where machines correspond to clusters, and the polytope constraints simply enforce that only one unit of CPU is allocated at any given time instant. In this single dimensional setting, PF is same as Weighted Round Robin. Using these facts, we obtain the alternate analysis as a corollary of Theorem 2. The sketch of this analysis can be found in Section 3. The original analysis in [12] relied on a dual-fitting argument.

2 Monotone Multi-cluster Polytope Scheduling

In this section we prove Theorem 2. First, we set up some notation. Recall that at every time instant, each alive job is assigned to exactly one cluster. Let A_{it} denote the set of alive jobs at time t that are assigned to cluster i in our algorithm's schedule. We often drop the subscripts t or i when it is clear from the context. Similarly, define A_t to be the set of all jobs alive at time t; that is, $A_t := \bigcup_i A_{it} = \{j \mid t \in [r_j, C_j]\}$. Let $\vec{y}_t$ denote the vector of processing rates of jobs. We use y_{jt} to denote the processing rate job j gets at time t. For any subset of jobs S, define $\vec{y}_t(S)$ as the projection of $\vec{y}$ into S; so, $y_{jt}(S) = y_{jt}$ if $j \in S$, and 0 otherwise.

2.1 Algorithm

Our algorithm for Monotone-MCPS consists of two components: Rate allocation using Proportional Fairness (PF) and job assignment using Selfish Migrate [12]

1. Proportional Fairness (PF): Each cluster i assigns rates to the set of jobs assigned to i that are alive at time t using the Proportional Fairness (PF) algorithm. The PF allocation can be obtained by solving the following convex program.

$$\text{Maximize} \quad \sum_{j \in A_{it}} w_j \log y_{jt} \qquad \text{s.t.} \qquad \vec{y}_t(A_{it}) \in \mathcal{P}_i \, .$$

   The rates assigned to jobs remain unchanged unless a new job arrives to the cluster or a job departs.

2. Selfish Job Migration: This rule is applied only when a job completes or arrives – at other times, no job changes its assignment. Selfish Migrate algorithm is best viewed as a game where each job tries to maximize its own utility (defined later). A key property of our assignment policy is that at each time instant, a job is assigned to the cluster that maximizes its utility. In other words, jobs are in Nash equilibrium with respect to their utility functions.

   To define the utility of a job, we need the notion of virtual queues. Each cluster has a complete *virtual* ordering of jobs assigned to the cluster, and the utility of a job depends on its position in this virtual ordering. We emphasize that this ordering is used only for the job assignment, and the PF algorithm itself is oblivious to this ordering. If job j is ahead of j' on cluster i, we denote it as $j \leq_i j'$. For a subset of jobs S and a job $j \in S$, let $y^*_{ij}(S)$ denote the rate PF assigns to job j if the set of jobs assigned to cluster i is S. Suppose a job j is on cluster i at time t. Then its utility on i is defined as $y^*_{ij}(A^{\leq j}_{it})$. The job's utility on any other cluster i' is defined as $y^*_{i'j}(A_{i't} \cup j)$. Now we describe how the virtual ordering of jobs are built on each cluster.

   - When a job j arrives at time t, it is assigned to the cluster i for which $y^*_{ij}(A_{it} \cup \{j\})$ is maximized. Further, the newly arrived job goes to the *tail of the virtual ordering.*
   - When a job completes, it simply disappears without affecting the relative ordering of the other jobs. However, this may start a chain of jobs migrations, as jobs may increase their utilities by switching to the cluster from which the job departed. Fix a job j. If job j is currently residing on cluster i, its utility is $y^*_{ij}(A^{\leq j}_{it})$. Here, $A^{\leq j}_{it}$ refers to all the jobs in A_{it} ahead of j in the virtual ordering including j itself. If job j moves to another cluster $i' \neq i$, the job is placed *behind* all jobs in $A_{i't}$ in the virtual ordering on i'. Hence, its utility will be $y^*_{i'j}(A_{i't} \cup j)$. A job j is free to move to any cluster i' as long as it's utility improves. If two jobs try to move simultaneously, then we break ties arbitrarily. This process is repeated until no jobs can improve their utilities. A priori, it is not clear if this process will terminate. For now we assume that it terminates, and in Section 2.4 we show that each jobs migrates at most $O(\frac{n}{\epsilon} \log n)$ times in total.

   This completes the description of our algorithm.

2.2 Key Properties Used in the Analysis

In this section, we summarize two key properties we will crucially use in our analysis. Recall that we assume in our problem Monotone-MCPS that all polytopes $\mathcal{P}_i$ are monotone. The following proposition is a restatement of Definition 1.

▶ **Proposition 3** (Polytope Monotonicity [11]). *Let $y^*_j(S)$ denote job j's processing rate under the PF algorithm for an arbitrary fixed monotone polytope $\mathcal{P}$. Then, for all $j \in S$ and $j' \notin S$, we have $y^*_j(S) \geq y^*_j(S \cup \{j'\})$.*

The next proposition, which we call the optimality condition, immediately follows from the convexity of the polytope and the PF algorithm's objective. We include the proof for completeness.

▶ **Proposition 4** (Optimality Condition [11]). *Let $\vec{y} \in \mathcal{P}$ denote any feasible rate vector for the jobs in S. If the space of feasible rates $\mathcal{P}$ is convex, then*

$$\sum_{j \in S} w_j \frac{y_j}{y^*_j(S)} \leq \sum_{j \in S} w_j \, .$$

Proof. For notational simplicity, let $y^*_j := y^*_j(S)$. Let $f(\vec{y}) = \sum_{j \in S} w_j \log y_j$. We have $\frac{\partial f(\vec{y^*})}{\partial y_j} = \frac{w_j}{y^*_j}$. The optimality of $\vec{y^*}$ implies $\nabla f(\vec{y^*}) \cdot (\vec{y} - \vec{y^*}) \leq 0$ for all $\vec{y} \in \mathcal{P}$. The proposition now follows by elementary algebra. ◀

These two conditions will be repeatedly used in our analysis. As mentioned earlier, our analysis is based on a potential function which depends on jobs arrival, and processing of our algorithm and the optimal scheduler. The potential changes will be categorized into two: discontinuous changes and continuous changes. As we will discuss soon in detail, discontinuous changes occur when jobs arrive or complete, and all other changes are continuous. Our analysis differs from the previous work in bounding discontinuous changes, and continuous changes due to the optimal scheduler's processing. In particular, the latter, which is formalized in Lemma 6, is the most interesting part in our analysis.

Before we move to the detailed analysis, we discuss at high level how we bound the continuous changes of the potential, particularly Lemma 6. As mentioned, the potential adds up the potential defined over each cluster, which only depends on the jobs assigned to the cluster by the algorithm and the optimal scheduler, which we denote A_i and O_i, respectively. If $A_i = O_i$ for all i, then the analysis is essentially equivalent to that of the single cluster case, which was done in [11]. Otherwise, using the greedy nature of the Selfish Migration rule and the Polytope monotonicity, we can w.l.o.g. proceed assuming that all jobs in O_i are also added to A_i for each cluster i. As a result, we are left with the task of upper bounding $\sum_{j \in O_i} w_j \cdot \frac{y^O_j}{y^*_j(A_i \cup O_i)}$; see Eqn. (4). At first sight, it is not clear how to bound this. The numerators are the processing rates of jobs in O_i due to the adversary, and the denominators are those under PF with extra jobs A_i added. This is where we apply the optimality condition, Proposition 4 assuming that PF is run on the jobs $A_i \cup O_i$ and setting y_j following the adversary for jobs $j \in O_i$; $y_j = 0$ for other jobs. Thus, the polytope monotonicity and the optimality condition are nicely combined to prove the key lemma.

2.3 Competitive Analysis: Proof of Theorem 2

We use amortized local competitiveness to prove the theorem. Our potential function $\Phi(t)$ is inspired by [6, 11]. The potential function adds up potential functions defined for individual clusters that are essentially identical to the potential in [11]. Define O_t and O_{it} for the optimal scheduler analogously as we did for A_t and A_{it} for our algorithm. For a set of jobs S, let $W(S)$ denote the total weight of jobs in the set. Assuming that our algorithm is given $(e+\epsilon)$-speed, we show that the following conditions are satisfied that will imply Theorem 2. These are standard conditions which are verified for most potential functions. See [13] for a tutorial on the framework.

1. (Boundary condition) $\Phi(0) = \Phi(\infty) = 0$;
2. (Discontinuous changes) Φ can only decrease when a job arrives into or departs from the system; and
3. (Continuous changes) At the other times t, $W(A_t) + \frac{d}{dt}\Phi(t) \leq \frac{3}{\epsilon^2} W(O_t)$.

It is an easy exercise to verify that conditions are sufficient to establish our algorithm's competitiveness by integrating the last inequality [13]. We give a brief explanation. Suppose all jobs are completed by our algorithm and OPT by time T. The first two conditions imply that $\int_{t=0}^{T} \frac{d}{dt}\Phi(t)dt \geq 0$. Then integrating the above inequality over time, we have:

$$\int_{t=0}^{T} W(A_t)dt + \int_{t=0}^{T} \frac{d}{dt}\Phi(t)dt \leq \frac{3}{\epsilon^2}\int_{t=0}^{T} W(O_t)dt\,.$$

This implies Theorem 2 since the first term above is the weighted flow time of our algorithm, and the RHS is that of OPT.

To define the potential function formally, we need to set up more notation. Fix a time instant t. For job j, let p_{jt} denote the remaining size of the job in the PF's schedule, and let p^O_{jt} denote the remaining size of the job in OPT's schedule. Define a job j's *lag* as $\tilde{p}_{jt} = \max(0, p_{jt} - p^O_{jt})$. The quantity $\tilde{p}_{jt}$ indicates how much our algorithm is behind the optimal schedule in terms of job j's processing. Let $L_t = \{j \in A_t \mid \tilde{p}_{jt} > 0\}$. Note that $A_t \setminus L_t \subseteq O_t$. Recall that $y^*_j(S)$ denote the optimal rate the PF algorithm allocates to job $j \in S$ when working on the set S. We define the following potential function:

$$\Phi(t) := \sum_i \Phi_i(t) \tag{2}$$

$$\text{where } \Phi_i(t) := \frac{1}{\epsilon}\sum_{j\in A_{it}} w_j \frac{\tilde{p}_{jt}}{y^*_{ij}(A^{\leq j}_{it})} \tag{3}$$

It now remains to verify all the above conditions(1-3) are satisfied. The boundary condition trivially holds since at times $t = 0$ and $t = \infty$, the algorithm has no alive jobs. In the following sections, we show the last two conditions hold true.

2.3.1 Discontinuous Changes

First we show that $\Phi(t)$ can only decrease when a job arrives or completes in our algorithm's schedule.

▶ **Lemma 5.** *The discontinuous changes in the potential function (2) due to a job arrival or departure is at most zero.*

Proof. Suppose a job j arrives at time t; for notational convenience, we assume that $j \notin A_t$. For the job j, $\tilde{p}_{jt} = 0$. Suppose j is assigned to cluster i. Since job j is behind any other jobs on the cluster in the virtual ordering of jobs, no existing terms in Φ_i change. A new term, $\left(w_j \cdot \frac{\tilde{p}_{jt}}{y^*_j(A_{it}\cup\{j\})}\right)$ is added for the job j, and the value of this term is 0 since $\tilde{p}_{jt} = 0$. Therefore, the lemma is true for job arrivals.

We now focus on job completions. It is easy to see that the optimal scheduler completing a job does not lead to any discontinuous changes in the potential function. Hence, we only consider changes in the potential due to our algorithm completing a job. Suppose a job j completes. If j was on cluster i just before it completed, the term $(w_j \cdot \frac{\tilde{p}_{jt}}{y^*_j(A^{\le j}_{it})})$ drops from $\Phi_i(t)$ as $\tilde{p}_{jt}$ becomes zero.

Due to this the value of other terms for jobs k such that $j \le_i k$ (that is, jobs that were behind the job j in the virtual order) may change from $(w_k \cdot \frac{\tilde{p}_{kt}}{y^*_j(A^{\le k}_{it})})$ to $(w_k \cdot \frac{\tilde{p}_{kt}}{y^*_j(A^{\le k}_{it}\setminus\{j\})})$. Such changes are non-positive due to the monotonicity of $\mathcal{P}_i$.

But completion of a job may result in a sequence of jobs migrations as other jobs may get a higher utility on the cluster that a job departed from. We show that the potential can only decrease when jobs migrate. Say a job j migrates from cluster i to cluster i'. Note that the term $(w_j \cdot \frac{\tilde{p}_{jt}}{y^*_j(A^{\le j}_{it})})$ drops from $\Phi_i(t)$, and a new term $(w_j \cdot \frac{\tilde{p}_{jt}}{y^*_j(A_{i't}\cup\{j\})})$ is added to $\Phi_{i'}(t)$.

When a job migrates from cluster i to i', the value of other terms for jobs k such that $j \le_i k$ change from $(w_k \cdot \frac{\tilde{p}_{kt}}{y^*_j(A^{\le k}_{it})})$ to $(w_k \cdot \frac{\tilde{p}_{kt}}{y^*_j(A^{\le k}_{it}\setminus\{j\})})$. This change in the value is non-positive due to the monotonicity of $\mathcal{P}_i$. Therefore, we have

$$\Delta\Phi_i(t) \le -\frac{1}{\epsilon} \cdot w_j \cdot \frac{\tilde{p}_{jt}}{y^*_j(A^{\le j}_{it})}.$$

Since j moves to cluster i', $(w_j \cdot \frac{\tilde{p}_{jt}}{y^*_j(A_{i't}\cup\{j\})})$ is added to $\Phi_{i'}(t)$. However, no terms in the summation of $\Phi_{i'}$ change since j is placed at the end in the ordering of jobs on i'. Hence we have

$$\Delta\Phi_{i'}(t) = \frac{1}{\epsilon} \cdot w_j \cdot \frac{\tilde{p}_{jt}}{y^*_j(A_{i't} \cup \{j\})}.$$

Since $\Phi(t)$ does not change on other cluster, we have $\Delta\Phi(t) = \Delta\Phi_i(t) + \Delta\Phi_{i'}(t) \le 0$, as desired. The inequality follows from the fact that job j having moved to i' means that $y^*_j(A^{\le j}_{it}) \le y^*_j(A_{i't} \cup \{j\})$. ◀

2.3.2 Continuous Changes

Fix a time instant t when no jobs arrive or depart. To simplify notation, we omit the subscript t from rest of the proof. Let $\Phi'|_O$ and $\Phi'|_A$ denote the potential changes due to OPT's processing and our algorithm's processing respectively. We note that $\frac{d}{dt}\Phi(t) = \Phi'|_A + \Phi'|_O$.

▶ **Lemma 6.** $\Phi'|_O \le \frac{1}{\epsilon}(W(A) + W(O))$.

Proof. Fix a cluster i and consider each job $j \in O_i$. We consider two cases. Let y^O_j denote the rate the optimal scheduler processes job j. Consider the case when j is also assigned to i by our algorithm at time t; that is, $j \in A_i$. Then, $\Phi'|_O$ due to job j is

$$w_j \cdot \frac{y^O_j}{y^*_j(A^{\le j}_i)} \le w_j \cdot \frac{y^O_j}{y^*_j(A_i \cup O_i)},$$

where the inequality follows due to PF being monotone for $\mathcal{P}_i$.

Next, we consider the other case where the algorithm processes job j on a different cluster i' from i. The reason the job j decided to stay on cluster i' instead of moving to cluster i is because it can't get a better utility when it is added to the end of the ordering of jobs on cluster i, i.e. $y_j^*(A_{i'}^{\leq j}) \geq y_j^*(A_i \cup \{j\}) \geq y_j^*(A_i \cup O_i)$. The last inequality is due to the monotonicity of $\mathcal{P}_i$ and the fact that $j \in O_i$. Hence $\Phi'|_O$ due to job j is

$$\Phi'|_O \leq w_j \cdot \frac{y_j^O}{y_j^*(A_{i'}^{\leq j})} \leq w_j \cdot \frac{y_j^O}{y_j^*(A_i \cup O_i)}.$$

Summing over all jobs O_i on each cluster i, we have

$$\Phi'|_O \leq \sum_i \sum_{j \in O_i} w_j \cdot \frac{y_j^O}{y_j^*(A_i \cup O_i)}. \tag{4}$$

Finally, we show

$$\sum_{j \in O_i} w_j \cdot \frac{y_j^O}{y_j^*(A_i \cup O_i)} \leq W(A_i) + W(O_i). \tag{5}$$

The above two equations 4 and 5 will yield $\Phi'|_O \leq \sum_i W(A_i) + W(O_i)$, proving the lemma.

To show equation (5), we appeal to the optimality condition stated in Proposition 4. Say we process jobs $A_i \cup O_i$ on cluster i using PF. Then each job j gets processed at a rate of $y_j^*(A_i \cup O_i)$. Then, set $y_j = y_j^O$ for all $j \in O_i$ and $y_j = 0$ for all other jobs. Note that this setting of $\{y_j\}$ is a feasible allocation of rates to jobs $A_i \cup O_i$. Hence, equation (5) follows from Proposition 4. ◀

We now bound Φ's continuous changes due to our algorithm's processing. Recall that $L = A \setminus O$ denote the set of jobs that the optimal scheduler has finished but are alive in PF schedule. Similarly, let $L_i = A_i \setminus O$.

We consider two cases.

Case 1: $W(L) \leq (1-\epsilon)W(A)$. Since $A \setminus L \subseteq O$, we have $W(O) \geq \epsilon W(A)$. Since $\Phi'|_A \leq 0$, we have:

$$W(A) + \Phi' \leq W(A) + \Phi'|_O \leq \frac{2}{\epsilon}(W(A) + W(O)) \leq \frac{3}{\epsilon^2} W(O)$$

where the second inequality follows from Lemma 6.

Case 2: $W(L) \geq (1-\epsilon)W(A)$. This is a more interesting case. If job j is on cluster i , then PF processes job j at a rate of $y_j^*(A_i)$. For every job $j \in L$, PF decreases $\tilde{p}_{jt}$ at the rate of $y_j^*(A_i)$. For all other jobs, PF can only decrease the potential. Hence we have,

$$\Phi'|_A \leq -\frac{1}{\epsilon} \sum_i \sum_{j \in L_i} w_j \frac{y_j^*(A_i)}{y_j^*(A_i^{\leq j})} \tag{6}$$

To bound this quantity, we use the following inequality which was shown in [11] using the polytope monotonicity and the optimality condition. For the sake of completeness, we repeat the proof in [11].

▶ **Lemma 7** ([11]). *Let S be an arbitrary ordered set of jobs. Let $S^{\leq j}$ denote jobs ahead of j, including j. For any subset $S' \subseteq S$, we have,*

$$\sum_{j \in S'} w_j \cdot \frac{y_j^*(S)}{y_j^*(S^{\leq j})} \geq W(S') \cdot \exp\left(-\frac{W(S)}{W(S')}\right).$$

Proof. For notational convenience, let $|S| = \kappa$, and number the jobs in S in increasing order of arrival time as $1, 2, \ldots, \kappa$. For $k > j$ and $k \leq \kappa$, let $\alpha_{jk} = \frac{y_j^*(S^{\leq k-1})}{y_j^*(S^{\leq k})}$. By the monotonicity of PF, we have $\alpha_{jk} \geq 1$. Define $\delta_{jk} = \alpha_{jk} - 1$. Note that $\delta_{jk} \geq 0$.

We now apply Proposition 4 to the set $\{1, 2, \ldots, k\}$ as follows: For jobs $j \in \{1, 2, \ldots, k\}$, the rate assigned by PF when executed on this set is $y_j^*(S^{\leq k})$, and this goes into the denominator in Proposition 4. We consider $y_j^*(S^{\leq k-1})$ for $j < k$, and $y_k^*(S^{\leq k-1}) = 0$ as a different set of rates that go into the numerator in Proposition 4. This yields:

$$\sum_{j=1}^{k-1} w_j \frac{y_j^*(S^{\leq k-1})}{y_j^*(S^{\leq k})} \leq \sum_{j=1}^{k} w_j \,.$$

Observing that $\frac{y_j^*(S^{\leq k-1})}{y_j^*(S^{\leq k})} = 1 + \delta_{jk}$, we obtain $\sum_{j=1}^{k-1} w_j \delta_{jk} \leq w_k$ for $k = 1, 2, \ldots, \kappa$.

Adding these inequalities for $k = 1, 2, \ldots, \kappa$ and changing the order of summations, we obtain:

$$\sum_{k=1}^{\kappa} \sum_{j=1}^{k-1} w_j \delta_{jk} = \sum_{j=1}^{\kappa} w_j \left(\sum_{k=j+1}^{\kappa} \delta_{jk} \right) \leq W(S) \,.$$

Hence,

$$\sum_{j \in S'} w_j \left(\sum_{k=j+1}^{\kappa} \delta_{jk} \right) \leq W(S) \,.$$

Let $\Delta_j = \sum_{k=j+1}^{\kappa} \delta_{jk}$, so that the above inequality becomes $\sum_{j \in S'} w_j \Delta_j \leq W(A)$. Now observe that

$$\frac{y_j^*(S)}{y_j^*(S^{\leq j})} = \prod_{k=j+1}^{\kappa} \frac{1}{\alpha_{jk}} = \prod_{k=j+1}^{\kappa} \frac{1}{1 + \delta_{jk}} \geq \exp\left(- \sum_{k=j+1}^{\kappa} \delta_{jk} \right) = \exp(-\Delta_j) \,.$$

We used the fact that $\delta_{jk} \geq 0$ for all j, k. Therefore,

$$\sum_{j \in S'} w_j \frac{y_j^*(S)}{y_j^*(S^{\leq j})} \geq \sum_{j \in S'} w_j \exp(-\Delta_j) \,.$$

Since $\sum_{j \in S'} w_j \Delta_j \leq W(S)$, the RHS is maximized when $\Delta_j = W(S)/W(S')$. Therefore,

$$\sum_{j \in S'} w_j \frac{y_j^*(S)}{y_j^*(S^{\leq j})} \geq \exp(-W(S)/W(S')) \sum_{j \in S'} w_j = W(S') \cdot \exp(-W(S)/W(S')) \,. \qquad \blacktriangleleft$$

By applying this lemma with $S' = L_i$ and $S = A_i$ for each machine i, we have,

$$\begin{aligned}
\epsilon\Phi'|_A &\leq -\sum_i \sum_{j \in L_i} w_j \frac{y_j(A)}{y_j(A^{\leq j})} && \text{[Equation 6]} \\
&\leq -\sum_i W(L_i) \exp\left(-\frac{W(A_i)}{W(L_i)}\right) && \text{[Lemma 7]} \\
&= -W(L) \sum_i \frac{W(L_i)}{W(L)} \exp\left(-\frac{W(A_i)}{W(L_i)}\right) \\
&\leq -W(L) \exp\left(-\sum_i \frac{W(L_i)}{W(L)} \cdot \frac{W(A_i)}{W(L_i)}\right) && \text{[Due to convexity of } \exp(-x)\text{]} \\
&= -W(L) \exp\left(-\frac{W(A)}{W(L)}\right) \\
&\leq -(1-\epsilon)W(A) \cdot \exp(-1/(1-\epsilon)) && \text{[Since } W(L) \geq (1-\epsilon)W(A)\text{]} \\
&\leq -\frac{1-2\epsilon}{e} \cdot W(A)
\end{aligned}$$

for $(0 \leq \epsilon < 1/2)$. Thus, when PF is given $(e + 3\epsilon)$ speed , we have $\Phi'|_A \leq -(1 + 1/\epsilon)W(A)$. This together with Lemma 6 gives,

$$W(A) + \Phi'|_A + \Phi'|_O \leq 1/\epsilon \cdot W(O)\,.$$

Thus we conclude that in both cases $W(A) + \Phi' \leq 3/\epsilon^2 W(O)$, completing the proof of Theorem 2.

2.4 Bounding the Number of Migrations

First, observe that jobs can't migrate forever. This is because the total utility of jobs strictly increases when a job migrates and there are only a finite number of configurations regarding where each job can be residing. To ensure the migration process ends in polynomial time, we allow each job moves to another cluster only when its utility increases by a factor of at least $(1 + \epsilon)$. Also we force job j to move to the cluster where its utility is maximized. For any polytope $\mathcal{P}_i$, it is well-known that a job gets a rate at least $1/n$ times the rate it would get when it is the only job on cluster i in the PF allocation. Let s_j be the maximum processing rate when j is the only job in the system. Then, it is easy to see that job j's processing rate/utility is at least s_j/n and at most s_j. Therefore, each job can migrate at most $O(\log_{1+\epsilon} n)$ times. Hence the total number of jobs migrations is at most $O(\frac{n}{\epsilon} \log n)$.

3 Non-Clairvoyant Scheduling On Unrelated Machines

As already mentioned, the unrelated machines model is a special case of Monotone-MCPS, where each machine corresponds to a cluster i. Recall that in the unrelated machine setting, if a job is assigned to machine i it takes p_j/s_{ij} time to complete. The term $0 \leq s_{ij} \leq 1$ is the machine dependent *slow-down* factor of job j. It is easy to verify that this can be captured using polytope constraints of the form $\sum_j x_{jt} \leq 1$ and $y_{jt} \leq s_{ij} \cdot x_{jt}$. Therefore, Theorem 2 gives an $(e + \epsilon)$-speed $O\left(1/\epsilon^2\right)$-competitive algorithm for minimizing the weighted flow-time in the non-clairvoyant setting. However, PF algorithm is the same as Weighted Round Robin (WRR) in the unrelated machine setting, hence by a more careful analysis of Lemma 7 we can reduce the speed augmentation to $2 + \epsilon$. The entire analysis of Theorem 2 also goes through if we replace WRR by the Latest Arrival Processor Sharing algorithm (LAPS) [5], and we obtain $(1+\epsilon)$-speed $O\left(1/\epsilon^2\right)$-competitive algorithm, matching the best known result [12]. This also gives the first analysis of LAPS for unrelated machine scheduling. The omitted details are simple, and we defer the complete proof to the full version of the paper.

References

1 Faraz Ahmad, Srimat T. Chakradhar, Anand Raghunathan, and T. N. Vijaykumar. Tarazu: optimizing mapreduce on heterogeneous clusters. In *ASPLOS*, pages 61–74. ACM, 2012. doi:10.1145/2150976.2150984.

2 Amazon EC2-Spot-Instances. URL: http://aws.amazon.com/ec2/spot-instances/.

3 Jivitej S. Chadha, Naveen Garg, Amit Kumar, and V. N. Muralidhara. A competitive algorithm for minimizing weighted flow time on unrelated machines with speed augmentation. In *STOC*, pages 679–684, 2009.

4 R. Cole, V. Gkatzelis, and G. Goel. Mechanism design for fair division: allocating divisible items without payments. In *ACM EC*, pages 251–268, 2013.

5 Jeff Edmonds and Kirk Pruhs. Scalably scheduling processes with arbitrary speedup curves. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 685–692, 2009.

6 Kyle Fox, Sungjin Im, and Benjamin Moseley. Energy efficient scheduling of parallelizable jobs. In *SODA*, pages 948–957, 2013.

7 N. Garg and A. Kumar. Better algorithms for minimizing average flow-time on related machines. In *ICALP (1)*, 2006.

8 A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, I. Stoica, and S. Shenker. Dominant resource fairness: Fair allocation of multiple resource types. In *NSDI*, 2011.

9 Robert Grandl, Ganesh Ananthanarayanan, Srikanth Kandula, Sriram Rao, and Aditya Akella. Multi-resource packing for cluster schedulers. In *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014*, pages 455–466, 2014. doi:10.1145/2619239.2626334.

10 Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. In *STOC*, 2014.

11 Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive flow time algorithms for polyhedral scheduling. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 506–524, 2015. doi:10.1109/FOCS.2015.38.

12 Sungjin Im, Janardhan Kulkarni, Kamesh Munagala, and Kirk Pruhs. Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 531–540, 2014. doi:10.1109/FOCS.2014.63.

13 Sungjin Im, Benjamin Moseley, and Kirk Pruhs. A tutorial on amortized local competitiveness in online scheduling. *SIGACT News*, 42(2):83–97, 2011. doi:10.1145/1998037.1998058.

14 Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *JACM*, 47(4):617–643, 2000.

15 Gunho Lee, Byung-Gon Chun, and Randy H Katz. Heterogeneity-aware resource allocation and scheduling in the cloud. In *Proceedings of the 3rd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud*, volume 11, 2011.

16 J. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.

17 Lucian Popa, Gautam Kumar, Mosharaf Chowdhury, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. Faircloud: sharing the network in cloud computing. In *ACM SIGCOMM*, pages 187–198, 2012.

18 Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. Improving mapreduce performance in heterogeneous environments. In *OSDI*, pages 29–42, Berkeley, CA, USA, 2008. USENIX Association. URL: `http://dl.acm.org/citation.cfm?id=1855741.1855744`.

19 S. M. Zahedi and B. C. Lee. REF: resource elasticity fairness with sharing incentives for multiprocessors. In *ASPLOS*, pages 145–160, 2014.

Revisiting Connected Dominating Sets: An Optimal Local Algorithm?*

Samir Khuller[1] **and Sheng Yang**[2]

1 **Dept. of Computer Science, University of Maryland, College Park, USA**
`samir@cs.umd.edu`

2 **Dept. of Computer Science, University of Maryland, College Park, USA**
`styang@cs.umd.edu`

Abstract

In this paper we consider the classical Connected Dominating Set (CDS) problem. Twenty years ago, Guha and Khuller developed two algorithms for this problem - a centralized greedy approach with an approximation guarantee of $H(\Delta)+2$, and a local greedy approach with an approximation guarantee of $2(H(\Delta)+1)$ (where $H()$ is the harmonic function, and Δ is the maximum degree in the graph). A local greedy algorithm uses significantly less information about the graph, and can be useful in a variety of contexts. However, a fundamental question remained - can we get a local greedy algorithm with the same performance guarantee as the global greedy algorithm without the penalty of the multiplicative factor of "2" in the approximation factor? In this paper, we answer that question in the affirmative.

1998 ACM Subject Classification G.2.2 Graph Algorithms

Keywords and phrases graph algorithms, approximation algorithms, dominating sets, local information algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.11

1 Introduction

A connected dominating set (CDS) in a graph is a subset of vertices that induces a connected subgraph, and is also a dominating set at the same time. A dominating set is a subset of vertices such that every node in the graph, is either in the dominating set, or adjacent to a node in the dominating set. Finding a minimum connected dominating set is NP-hard, and thus for the last twenty years, researchers have explored approximation algorithms for this problem starting with the work of Guha and Khuller [6][1]. One of the original motivations for the problem was in building a backbone for routing in the context of wireless ad hoc networks. Over time many other applications have been explored in the context of social networks and AI [2, 10, 3].

In their paper, Guha and Khuller [6] developed two simple approaches - the first one is a "local" approach, where we start from a single vertex in the solution, and incrementally (greedily) add neighboring nodes while maintaining a connected subset of nodes at all times. It is tempting to imagine that adding one node at a time, maintaining connectivity might work well. However, this does not work and there are instances where we might end up with a solution with $\Omega(n)$ nodes, while the optimum solution has only $O(1)$ nodes! Interestingly, a simple modification of this algorithm that explores a 2-hop neighborhood

* This work is supported by NSF grant CCF 1217890.
[1] This work was recently awarded the ESA Test of Time Award.

making greedy choices at each step that involves selecting *upto two* nodes at each step, works much better and they show that a $2(H(\Delta)+1)$ approximation can be obtained for this problem ($H(n) = \sum_{i=1}^{n} \frac{1}{i}$ is harmonic function, and Δ is the largest degree in the graph). The main benefits of the local algorithm are that no knowledge of the entire graph is needed at each step, and only the part of the "explored" graph suffices to select the next node. This may not be a useful for a static graph, as you can calculate it once and forever. But to get a CDS for a graph that changes dynamically, like social networks or wireless ad hoc networks, limiting the amount of information required can be both important and challenging.

They also developed an improved global algorithm that is again a greedy algorithm, and constructs a solution that does not maintain any connectivity property, and is only connected at the very end. This centralized greedy algorithm yields a bound of $H(\Delta)+O(1)$, eliminating the factor of 2, and gets us close to the lower bound on this problem of $(1-\epsilon)H(\Delta)$ due to set cover hardness [5]. Despite much interest (see book by Du and Wan [3]), the key question remained open for two decades - is there a local algorithm that gives us the same bound as the global algorithm?

In this paper, we answer this question in the affirmative. We develop a very simple local algorithm, and are able to show that it matches the bound of the global algorithm. Such a result is especially surprising due to its simplicity.

Connected Dominating Sets became a central topic in the context of wireless ad hoc networks, where the CDS acts as a routing backbone for packet routing. Often it is expensive to connect all the nodes, as the cost can become prohibitive, and in this case it is fine to connect *most* of the nodes (or a given fraction). Liu and Liang [9] formalized the problem of partial connected dominating sets (PCDS) and provided heuristics without performance guarantees. Avrachenkov et al. [1] defined the budgeted connected dominating set problem (BCDS) where we have a budget of k nodes and we wish to find a connected set of at most k nodes that maximizes the number of nodes it can cover. Inspired by applications in social networks, they developed practical heuristics using only local information. Khuller et al. [8] developed the first algorithms with theoretical guarantees for both these problems with approximation factors of $O(\ln \Delta)$ for PCDS and $\frac{1}{13}\left(1-\frac{1}{\epsilon}\right)$ for BCDS.

Extensions to node weighted versions were considered by Guha and Khuller [7] as well. Extensive research was subsequently done on this topic with the development of distributed algorithms [4], as well as for many special classes of graphs [3].

1.1 Our Contributions

Our results can be summarized as follows.

- In Section 3, for the Connected Dominating Set problem, we obtain the first local information algorithm whose approximation ratio is within additive constant to global algorithm, i.e. $H(\Delta)+O(1)$. To be precise, our approximation guarantee is $H(2\Delta+1)+1$. This algorithm requires 2-hop local information (see Section 2 for definition).
- In Section 4, with 1-hop local information, we obtain an $H(\Delta)+2\sqrt{H(\Delta)}+1$ approximation algorithm. In addition to better approximation ratio, it also runs faster than the Randomized CDS algorithm [6] (Section 2.3), because it explores fewer nodes.

2 Background

We first review existing approaches [6, 2].

2.1 Global Algorithm for CDS

The global algorithm runs in two phases. Initially, all nodes are colored white. In the first phase, the algorithm iteratively adds a node to the solution, colors it black and all its adjacent white nodes gray. A *piece* is defined as a white node or a black connected component. A new node is chosen to be colored black to get maximum reduction in the number of pieces. This phase ends when no such node exists that can give non-zero reductions. At this time, there are no white nodes left. Intuitively speaking, black nodes are selected nodes, gray nodes are nodes that are dominated, i.e. adjacent to black nodes.

In the second phase, we start with a dominating set that consists of several black components that we need to connect. The connection is done by recursively connecting pairs of black components with a chain of vertices, until there is only one black component, which will be our final solution.

The approximation ratio for this algorithm is $H(\Delta)+2$, where $H(n)$ is harmonic function.

2.2 2-hop Local Information Algorithm for CDS

The same paper proposed another algorithm, using only local information. Instead of using information of the entire graph, it only relies on information within 2-hops to the nodes chosen in the solution. The formal definition of local information is as follows.

Before we define what local information is, we first define the distance between a node and a set of nodes.

▶ **Definition 1** (Distance). In undirected graph, denote the distance between u and v in a graph as $d(u,v)$. It is the length of shortest path from u to v. $d(u,S)$ is defined to be $\min_{v\in S} d(u,v)$

We now define the local neighborhood of some node, or a set of nodes.

▶ **Definition 2** (Local Neighborhood). Given a set of nodes S in graph G, the r-hop neighborhood around S is the induced subgraph of G containing all nodes v such that $d(v,S) \leq r$. We denote the r-hop neighborhood as $N^r(S)$. When there is no confusion, we use the same notation to denote the set $N^r(S) = \{v|d(v,S) \leq r\}$.

An algorithm with local information uses information only within the local neighborhood of the nodes it has chosen. To be specific, if at some step, the set of nodes that an algorithm has chosen is S, and we have r-hop local information, then we know the induced graph of $N^r(S)$, as well as the degree of nodes in $N^r(S)/N^{r-1}(S)$.

From an arbitrary starting node, nodes are added iteratively. For each loop in this algorithm, one chooses a node, or a node and one of its neighbors. This means we need knowledge of 2-hop neighborhood to maximize the number of newly covered nodes, which explains why it uses 2-hops of local information.

The approximation ratio for this algorithm is $2(H(\Delta)+1)$. We can improve it in practice by maximizing the number of newly covered nodes for each new node selected, which is used in [10], but the theoretical bound is the same.

2.3 1-hop Local Information Algorithm for CDS

Borgs et al. [2] first came up with this algorithm, which is based on the previous one. Instead of choosing a node or a pair of adjacent nodes greedily, it chooses only one gray node to maximize the number of newly covered nodes, and in addition selects one of the newly covered

Algorithm 1: CDS with 2-hop Local Information

```
Data: Graph $G = (V, E)$
Result: A connected dominating set $S$
1 $s \leftarrow$ an arbitrary node;
2 $S \leftarrow \{s\}$;
3 while $S$ is not a dominating set do
4     $\bar{S} \leftarrow$ a node $u$ in $N^1(S)$ or a node in $N^1(S)$ and one of its neighbors in $N^1(u)$
       (which also lies in $N^2(S)$) that maximize the number of newly covered nodes;
5     $S \leftarrow S \cup \bar{S}$;
```

Algorithm 2: Randomized CDS with 1-hop Local Information

```
Data: Graph $G = (V, E)$
Result: A connected dominating set $S$
1 $s \leftarrow$ an arbitrary node;
2 $S \leftarrow \{s\}$;
3 while $S$ is not a dominating set do
4     $v \leftarrow$ a node in $N^1(S)$ that maximize the number of newly covered nodes;
5     $u \leftarrow$ a uniformly randomly chosen node from $N^1(v) - N^1(S)$, i.e. the newly
       covered nodes;
6     $S \leftarrow S \cup \{u, v\}$;
```

nodes uniformly at random. The maximization process only requires 1-hop neighborhood information, and we do not worry about the random node we are about to choose. Not only does this algorithm require less information, it runs much faster.

The approximation ratio for this algorithm is $2(H(\Delta) + 1)$.

2.4 Inspiration

Comparing the two algorithms using global information and local information, we find a gap of factor 2 in the approximation ratio: $2(H(\Delta)) + O(1)$ for a local algorithm versus $H(\Delta) + O(1)$ for global algorithm. This looks reasonable: we are always better off when offered more information. But is it really the case? Is this gap the price we paid for lack of information essential, or the same approximation ratio can be achieved regardless of limited information. Our answer is affirmative: with local information, we can still get $H(\Delta) + O(1)$ approximation. To be specific, an $H(2\Delta + 1) + 1$ approximation.

3 Improved 2-Hop Local Information Algorithm

3.1 Algorithm

S is initially any node. Iteratively add nodes within $N^2(S)$ to S. When new nodes are added to S, the 2-hop neighborhood of S expands, and we repeat this process. By the end of the process, we have colored all the nodes. Initially, all nodes are white. When a node is added to S, we color it black, and all its white neighbors gray. The selected black nodes will form components, and these components may merge when additional nodes are selected.

At each step, we look within a 2-hop neighborhood of S, i.e., among the nodes that are either gray or adjacent to gray nodes. Add the node v that maximizes $2w_v + c_v - 1$, where c_v

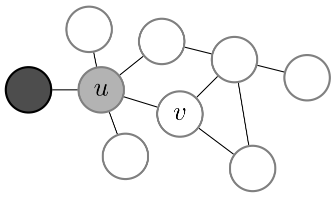


Figure 1 Consider u and v as potential choices. For node u, $c_u = 1$, as there is only adjacent component. Note that $w_u = 4$, it has four white neighbors, and itself is not white, hence $2 \cdot w_u + c_u - 1 = 8$. For node v, since it is white, there is no adjacent black component, so $c_v = 0$. Note w_v equals 3, since both node v itself and two of its neighbors are white. So the value is $2 \cdot w_v + c_v - 1 = 6 + 0 - 1 = 5$.

Algorithm 3: CDS with 2-hop Local Information

```
  Data: Graph $G = (V, E)$
  Result: A connected dominating set $S$
1 $s \leftarrow$ an arbitrary node;
2 $S \leftarrow \{s\}$;
3 $\bar{V} \leftarrow N^2(S)$;
4 while $\exists v \in \bar{V}$, s.t. $2w_v + c_v - 1 > 0$ do
5     $v \leftarrow \text{argmax}_{v \in \bar{V}} 2w_v + c_v - 1$;
6     $S \leftarrow S \cup \{v\}$;
7     $\bar{V} \leftarrow N^1(N^1(v))$;
8     for all nodes in $V$, update $c_v, w_v$;
```

is the number of different black components connected to v, and w_v is the number of white nodes in $N^1(v)$. Our knowledge of the graph is enriched when we add nodes to our solution. The algorithm ends when there is no v such that $2w_v + c_v - 1 > 0$. Note that if v was gray before the selection, c_v is guaranteed to be greater or equal to 1. If it was white, $c_v = 0$.

To make everything clear, here is the pseudo code.

3.1.1 Correctness

First we prove that what we get is a dominating set. If not, then there exists a node that is not dominated, i.e. a white node u. But $2w_u + c_u - 1 = 2w_u + 0 - 1 \geq 2 - 1 > 0$, we would have added this node to our solution, a contradiction.

Next, we prove the following theorem that this dominating set is indeed connected.

▶ **Theorem 3.** *The solution returned by Algorithm 3 is connected.*

We have the following observation:

▶ **Observation 4.** *When a node is added to our solution, it is within 2-hops of some black node.*

It is true because otherwise, this node would be beyond our knowledge and would not be considered at this step. This ensures the following corollary:

▶ **Corollary 5.** *Each newly added node can be connected to existing components at the cost of at most one additional node.*

Proof of Theorem 3. We prove this by contradiction. Suppose the algorithm gives a dominating set that is not connected, i.e. has more than one component. Exactly one of these components will contain the starting node, call it starting component. Consider the first time when a node u outside the starting component joins our solution. According to Corollary 5, there must exist another node v that can join u to the starting component. Since u is disconnected from the starting component, v is not in our solution. But at the end of the algorithm, $2 \cdot w_v + c_v - 1 \geq 0 + 2 - 1 > 0$, which means that v can be added to our algorithm. This gives a contradiction. ◀

So our algorithm will indeed give a connected dominating set.

3.2 Analysis

We are going to use a charging scheme to charge the cost of each selected node and bound the total charge, which in turn bounds the number of nodes we select. Before that, we define uncharged black node as:

▶ **Definition 6.** An uncharged black node is a node that was charged once when it turned directly from white to black, and has not been charged afterwards.

For uncharged black nodes, the following lemma holds.

▶ **Lemma 7.** *Each component contains exactly one uncharged black node.*

The proof of this lemma will come later.

Recall in the algorithm, we select a node v that maximizes $2w_v + c_v - 1$, where w_v is the number of white nodes in $N^1(v)$, and c_v is the number of black components adjacent to v. We charge 1 for selecting this node, and split this charge into $2w_v + c_v - 1$ shares. For every white neighbor u of v, it takes two shares, i.e. gets $2/(2w_v + c_v - 1)$ charge. For all but one adjacent components, a share of charge is placed on the uncharged black node of this component (assuming the correctness of Lemma 7). Notice v may be white or gray before the selection. If it was white, it means that v is not adjacent to any black nodes, so $c_v = 0$, and it is charged one share, i.e. $1/(2w_v + c_v - 1)$. If it was gray, then nothing needs to be done. In conclusion, if the node was white, the number of shares equals $2(w_v - 1) + 0 + 1 = 2w_v + c_v - 1$; if the node was black, the number of shares equals $2w_v + (c_v - 1) = 2w_v + c_v - 1$. This means we are not over or under counting charges. A visual explanation is in Figure 2.

We now prove the correctness of Lemma 7.

Proof of Lemma 7. An uncharged black node comes into existence when a white node is chosen and added to our solution. According to the charging scheme, this node itself forms a component, and it has exactly one uncharged black node.

Components are connected when a gray node is chosen which connects several components. Since all but one component was charged, assuming all existing components have exactly one uncharged black node, the resulting component also has exactly one uncharged black node. A visual explanation is in Figure 3. ◀

Everything prepared, we state the main theorem and prove it by bounding the total charge.

▶ **Theorem 8** (Main Theorem for 2-hop Local Information Algorithm)**.** *The improved 2-hop local information algorithm gives $H(2\Delta + 1) + 1$ approximation.*

Proof of Theorem 8. Suppose the optimal solution is OPT, $|\text{OPT}| = k$, with vertices v_{OPT_1}, v_{OPT_2}, ..., v_{OPT_k}. We partition all nodes into S_{OPT_1}, S_{OPT_2}, ..., S_{OPT_k}. Without loss of

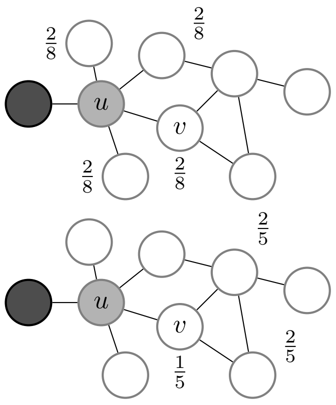


Figure 2 Consider the charging for u and v if they were selected at this step. For u, $2 \cdot w_u + c_u - 1 = 8$, each white neighbor of u gets 2 shares. For node v, $2 \cdot w_v + c_v - 1 = 6 + 0 - 1 = 5$. Each white neighbor gets 2 shares, but since node v goes from white directly to black, it only gets one share, and become an uncharged black node.

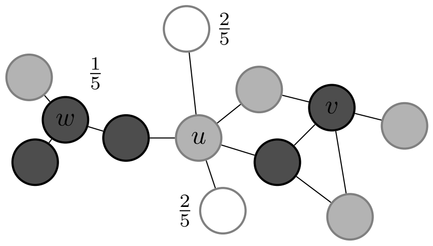


Figure 3 Suppose the black nodes are already chosen, and we are adding node u to the solution. There are two black components adjacent to u, so $c_u = 2$. Thus $2w_u + c_u - 1 = 4 + 2 - 1 = 5$. Each white neighbor of u gets 2 shares. For the two components, all but one component will get a share. This share is charged against the uncharged black node (node w for the left component and node v for the right component) of the component, which may not be the node adjacent to u. After charging, the two components are merged, and the merged component still has exactly one uncharged black node, i.e. node v. Note there is no charge against node u.

generality, we reorder the nodes, and use i to denote vertex v_{OPT_i}. So S_i contains vertex i and its neighbors. Ties are broken arbitrarily as long as $i \in S_i$.

Without loss of generality, we assume that the charge in S_i changes at each step. To describe the total number of charges that nodes in S_i can receive, we define p_i^j as the following value for S_i in step j:

$$2 \cdot w_i^j + b_i^j - 1$$

where w_i^j is the number of white nodes in S_i at step j, b_i^j is the number of uncharged black nodes at step j in S_i. A symbol (e.g. w_i, b_i) with an upper script j, i.e. w_i^j, b_i^j, c_i^j, denotes the corresponding value at step j. $p_i^1 = 2\delta(i) + 1$ ($\delta(i)$ is the degree of node i). A white node can receive two charges (in fact all white nodes will receive two charges, except for one, which is the only uncharged black node when the algorithm ends). Either it will receive two charges when it first becomes gray, and never receive any more charge. Or it receives one charge when it becomes a uncharged black node, and receives another one to become a charged black node. We have the following lemma:

▶ **Lemma 9.** *When the total charge inside S_i changes, p_i^j decreases by at least one.*

Proof of Lemma 9. Total charge changes when some node in S_i receives charges. There can be three cases: the node was white, gray, or black. If this node was white before charging, either it is now gray, which means it receives two charges, or it is now black, meaning one charge. And p_i^j will decrease by 1 or 2. If it was gray, it must have been fully charged and will not receive any more charge. If it was black, it must have been an uncharged black node to receives one charge. In this case, p_i^j also decrease one. For all three cases, either there is no charge change in S_i, or there is a decrease on p_i^j of at least 1. ◀

We use p^j instead of p_i^j when there is no confusion. Notice $b_i^j \leq c_i^j$, since every uncharged black node corresponds to a component, but the uncharged black node may not be in S_i. Consider each step. Suppose the selected node is v . Since node i is also a valid choice, we have

$$2w_v^j + c_v^j - 1 \geq 2w_i^j + c_i^j - 1 \geq 2w_i^j + b_i^j - 1 = p^j$$

The number of charges that S_i receives at step j is $p^j - p^{j+1}$, so the total charge in this step is:

$$\frac{p^j - p^{j+1}}{2w_v^j + c_v^j - 1} \leq \frac{p^j - p^{j+1}}{p^j}$$

This holds when $2w_i^j + c_i^j - 1 > 0$, which is guaranteed to be true when $p_i^j > 0$. The inequality will break down when $w_v^j = 0$ and $c_v^j = 1$, i.e. when $2w_i^j + c_i^j - 1 = 0$. To fix it, we notice $\forall S_i, \exists k_i, s.t. p_i^{k_i} > 0, \forall t > k_i, p_i^t \leq 0$. Thus we can take out the last term and bound it separately. So the total charge until step k_i (including step k_i) is upper bounded by:

$$\begin{aligned}
\sum_{j=1}^{k_i} \frac{p^j - p^{j+1}}{p^j} &= \sum_{j=1}^{k_i} \sum_{t=p^{j+1}+1}^{p^j} \frac{1}{p^j} \\
&\leq \sum_{j=1}^{k_i-1} \sum_{t=p^{j+1}+1}^{p^j} \frac{1}{t} + \left(\sum_{p^{k_i+1}+1}^{\max\{p^{k_i+1}+1,1\}-1} 1 + \sum_{t=\max\{p^{k_i+1}+1,1\}}^{p^{k_i}} \frac{1}{t} \right) \\
&\leq \left(\sum_{j=1}^{k_i-1} \sum_{t=p^{j+1}+1}^{p^j} \frac{1}{t} + \sum_{t=\max\{p^{k_i+1}+1,1\}}^{p^{k_i}} \frac{1}{t} \right) + \sum_{p^{k_i+1}+1}^{\max\{p^{k_i+1}+1,1\}-1} 1 \\
&= \sum_{j=1}^{p^1} \frac{1}{j} + \sum_{p^{k_i+1}+1}^{\max\{p^{k_i+1}+1,1\}-1} 1 \\
&= H(p^1) + (0 - p_i^{k_i+1})
\end{aligned}$$

where $H(n) = \sum_{i=1}^{n} \frac{1}{i}$ is harmonic sum. The last equality uses the fact that $2w_v + c_v - 1 \geq -1$.

Using $p_i^{k_i+1} \leq 0$, $p_i^t \geq -1$, combined with our assumption that p_i^j decreases at each step, there is at most one more step before the whole algorithm stops. So the total amount of charge is upper bounded by:

$$(p_i^{k_i+1} - p_i^{k_i+2}) \cdot 1 \leq (p_i^{k_i+1} + 1)$$

Adding together, we have,

$$H(p^1) + (0 - p_i^{k_i+1}) + (p_i^{k_i+1} + 1) = H(p^1) + 1$$

As $p^1 = 2w_i + b_i - 1 = 2w_i - 1 \leq 2\Delta + 1$, the total charge in S_i is bounded by $H(2\Delta + 1) + 1$. Since there are $|OPT|$ different S_i, the total charge is bounded by $|OPT|(H(2\Delta + 1) + 1)$, which is also the upper bound for the number of nodes we choose. ◀

Algorithm 4: Improved Algorithm for CDS with 1-hop Local Information

```
   Data: Graph $G = (V, E)$
   Result: A connected dominating set $S$
1  $s \leftarrow$ an arbitrary node;
2  $S \leftarrow \{s\}$;
3  while $S$ is not a dominating set do
4      $d \leftarrow$ the largest degree in $N^1(S)$;
5      $p \leftarrow \frac{1}{\sqrt{H(d)}}$;
6      $v \leftarrow$ a node in $N^1(S)$ that maximizes the number of newly covered nodes;
7      $S \leftarrow S \cup \{v\}$;
8      if with probability $p$ then
9          $u \leftarrow$ a node from the newly covered nodes uniformly at random;
10         $S \leftarrow S \cup \{u\}$;
```

4 Improved 1-Hop Local Information Algorithm

4.1 Intuition

Recall that the algorithm by Borgs et al. [2], which selects a random neighbor when a gray node is chosen. It is too expensive to select a random node every time. If instead of picking a random neighbor every time, we can pick it with some probability p. The total approximation ratio will be (this is only an intuition, detailed proof is in Section 4.3) $(1+p)(\frac{1}{p} + H(\Delta))$ that can be minimized when $p = \frac{1}{\sqrt{H(\Delta)}}$. This works given the assumption that Δ is known before hand, which is not always the case.

4.2 Algorithm

Instead of using the largest degree in the graph, every time when calculating p, we use the largest degree in the explored graph. In another world, the largest degree in $N^1(S)$. Below is the pseudocode.

4.3 Analysis

Like the previous analysis, we do charging, and partition all the nodes into $S_1, S_2, \ldots, S_{OPT}$ in the same manner. We bound the total charge inside S_i, and the total number of nodes chosen in our solution that corresponds to these charges.

▶ **Theorem 10** (Main Theorem for 1-hop Local Information Algorithm). *The improved 1-hop local information algorithm gives $H(\Delta) + 2\sqrt{H(\Delta)} + 1$ approximation.*

Proof of Theorem 10. When a node is selected in line 6 in Algorithm 4, we charge 1, and the charge is uniformly divided among all the newly covered nodes. In line 9, another node is chosen with probability $p = \frac{1}{\sqrt{H(d)}}$, where d is the largest degree we currently know. Thus whenever S_i receives charge c at some step, the expected number of nodes in our solution increases by $c(1+p)$. Note that this p changes over time.

The charging for each S_i is divided into two phases. The first phase ends when one of the nodes in S_i got chosen. In another word, we come to the second phase when v_i is visible. For the first phase, we model the charging process with the following problem.

▶ **Definition 11.** For $1 \leq j \leq n$, let X_j, Y_j be independent Bernoulli random variables with $E[X_j] = p_j \in [0,1], E[Y_j] = p'_j \in [0,1]$. Let T be the random variable denoting the smallest j such that $X_jY_j = 1$ (or n if $X_jY_j = 0$ for all j).

Here X_j is a random variable indicating whether a node in S_i was chosen in the j-th step. Y_j indicate whether a random neighbor was chosen at this step. We will prove the following theorem

▶ **Theorem 12.**

$$\mathbb{E}_T[\sum_{j=1}^{T}(X_j + X_jY_j)] \leq 1 + \sqrt{H(\Delta)}\,.$$

Proof. We prove it by induction. If $T = 0$, it is trivial. Suppose it holds for $T = t-1$, we prove it holds for t

$$\begin{aligned}
&\mathbb{E}_T\left[\sum_{j=1}^{T}(X_j + X_jY_j)\right] \\
=&p_1p'_1(1+1) + p_1(1-p'_1)\left(1 + \mathbb{E}_T\left[\sum_{j=2}^{T}(X_j + X_jY_j)\,|X_1 = 1, Y_1 = 0\right]\right) \\
&+ (1-p_1)\mathbb{E}_T\left[\sum_{j=2}^{T}(X_j + X_jY_j)\,|X_1 = 0\right] \\
=&2p_1p'_1 + p_1(1-p'_1) + p_1(1-p'_1)\mathbb{E}_T\left[\sum_{j=2}^{T}(X_j + X_jY_j)\right] \\
&+ (1-p_1)\mathbb{E}_T\left[\sum_{j=2}^{T}(X_j + X_jY_j)\right] \\
=&p_1 + p_1p'_1 + (1-p_1p'_1)\mathbb{E}_T\left[\sum_{j=2}^{T}(X_j + X_jY_j)\right] \\
\leq&p_1 + p_1p'_1 + (1-p_1p'_1)(1+\sqrt{H(\Delta)}) \\
=&p_1 + 1 + (1-p_1p'_1)\sqrt{H(\Delta)} \\
=&1 + \sqrt{H(\Delta)} + p_1(1 - p'_1\sqrt{H(\Delta)}) \\
\leq&1 + \sqrt{H(\Delta)} + p_1(1 - \frac{1}{H(\Delta)}\cdot\sqrt{H(\Delta)}) \\
=&1 + \sqrt{H(\Delta)}
\end{aligned}$$

◀

The last inequality uses the fact that $p'_j = \frac{1}{\sqrt{H(\delta(v))}}$ for some node v, and $\delta(v) \leq \Delta$. So $p'_j \geq \frac{1}{H(\Delta)}$

As for the second phase, whenever S_i receives charge c, the expected number of nodes increases by $c(1+p)$, $p = \frac{1}{\sqrt{H(d)}}$, where d is the largest degree we currently know. Since $d_i = \delta(v_i)$ is already known, we have $d \geq d_i$. So $p = \frac{1}{\sqrt{H(d)}} \leq \frac{1}{\sqrt{H(d_i)}}$, which implies that the increase in expected number of nodes in S_i is bounded by $\frac{c}{\sqrt{(H(d_i))}}$

The total charge is bounded by $H(d_i)$. This can be done using the standard technique in set cover. Thus the total number of nodes chosen in phase 2 is bounded by

$$\begin{aligned} H(d_i)\,(1+p) \leq& H(d_i)\left(1+\frac{1}{\sqrt{H(d_i)}}\right) = H(d_i) + \frac{H(d_i)}{\sqrt{H(d_i)}} = H(d_i) + \sqrt{H(d_i)} \\ \leq& \sqrt{H(\Delta)} + H(\Delta) \end{aligned}$$

Combining the charge from both phases, the expected number of nodes chosen in S_i is bounded by $H(\Delta)+2\sqrt{H(\Delta)}+1$. This directly means that expected the size of solution is bounded by $|OPT| \cdot (H(\Delta)+2\sqrt{H(\Delta)}+1)$, implying $H(\Delta)+2\sqrt{H(\Delta)}+1$ approximation. ◀

5 Future work

With only local information, we get the same approximation ratio as when we have global information, for connected dominating set problem. Is it also the case for other problems? Or does the lack of information prove to be a huge obstacle for designing algorithms?

Our first algorithm requires information within 2-hops. When only 1-hop local information is available, we cannot get the same result. Compared with previous result, the speed and approximation ratio is improved, i.e. $(H(\Delta)+2\sqrt{H(\Delta)}+1)$. But a gap still persists. Can we do better? Or is this the price we pay for lack of information?

References

1. Konstantin Avrachenkov, Prithwish Basu, Giovanni Neglia, Bernardete Ribeiro, and Don Towsley. Pay few, influence most: Online myopic network covering. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 813–818. IEEE, 2014.
2. Christian Borgs, Michael Brautbar, Jennifer Chayes, Sanjeev Khanna, and Brendan Lucier. The power of local information in social networks. In *Internet and Network Economics*, pages 406–419. Springer, 2012.
3. Ding-Zhu Du and Peng-Jun Wan. *Connected dominating set: theory and applications*, volume 77. Springer Science & Business Media, 2012.
4. Devdatt Dubhashi, Alessandro Mei, Alessandro Panconesi, Jaikumar Radhakrishnan, and Aravind Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 717–724. Society for Industrial and Applied Mathematics, 2003.
5. Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.
6. Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.
7. Sudipto Guha and Samir Khuller. Improved methods for approximating node weighted steiner trees and connected dominating sets. *Information and computation*, 150(1):57–74, 1999.
8. Samir Khuller, Manish Purohit, and Kanthi K Sarpatwar. Analyzing the optimal neighborhood: algorithms for budgeted and partial connected dominating set problems. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1702–1713. SIAM, 2014.

9 Yuzhen Liu and Weifa Liang. Approximate coverage in wireless sensor networks. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 68–75. IEEE, 2005.

10 Adish Singla, Eric Horvitz, Pushmeet Kohli, Ryen White, and Andreas Krause. Information gathering in networks via active exploration. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 981–988. AAAI Press, 2015.

Online Energy Storage Management: an Algorithmic Approach

Anthony Kim[1], Vahid Liaghat[2], Junjie Qin[3], and Amin Saberi[4]

1 Stanford University, Stanford, CA, USA
tonyekim@stanford.edu
2 Stanford University, Stanford, CA, USA
vliaghat@stanford.edu
3 Stanford University, Stanford, CA, USA
jqin@stanford.edu
4 Stanford University, Stanford, CA, USA
saberi@stanford.edu

Abstract

Motivated by the importance of energy storage networks in smart grids, we provide an algorithmic study of the online energy storage management problem in a network setting, the first to the best of our knowledge. Given online power supplies, either entirely renewable supplies or those in combination with traditional supplies, we want to route power from the supplies to demands using storage units subject to a decay factor. Our goal is to maximize the total utility of satisfied demands less the total production cost of routed power. We model renewable supplies with the zero production cost function and traditional supplies with convex production cost functions. For two natural storage unit settings, private and public, we design poly-logarithmic competitive algorithms in the network flow model using the dual fitting and online primal dual methods for convex problems. Furthermore, we show strong hardness results for more general settings of the problem. Our techniques may be of independent interest in other routing and storage management problems.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Online Algorithms, Competitive Analysis, Routing, Storage, Approximation Algorithms, Power Control

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.12

1 Introduction

With recent advancements in renewable energy generation technologies and smart grids, the problem of energy storage management has become a central problem. While renewable energy sources such as solar and wind are expected to supply a significant portion of electricity demand (by some measure, 50% by 2050 [37, 19, 23]), they have rather intermittent and variable output compared to the traditional fossil-fuel power generators. These uncertainties can lead to supply-demand imbalance and higher reserve requirements and pose a significant challenge to the renewable power supplies' integration to the existing power grids and energy distribution to consumers.

Energy storage provides a solution for maintaining supply-demand balance by providing the flexibility of transporting energy across time, just as a power network provides transportation flexibility over a geographical area. Many storage technologies have been researched and developed: batteries, flywheels, pumped-hydro, and compressed air energy storages [27, 17].

These technologies can help integrate renewable energy sources, such as solar arrays and wind farms, to the power grids as they become more robust, reliable and economically viable.

In recent years, the energy storage management problem has become a focus of active research by the power systems community. The problem of single storage control has been investigated extensively in various settings and several analytical solutions using stochastic dynamic programming exist (e.g., [34, 39, 35, 21]). For the general case when multiple storage units are available on a power network, analytical solutions with efficient algorithmic implementation are more challenging to obtain. We mostly have heuristics such as Model Predictive Control [41] and look-ahead policies [32] without any performance guarantees with the exception of few cases (e.g., [36] for a long-term average cost minimization problem).

Motivated by the importance of energy storage networks within smart grids, we provide an algorithmic study of an online problem of energy storage management with storage units subject to decay. We consider both private and public storage unit settings and model the unpredictable output of renewable sources as online power supplies and the predictable, say hourly, variation of demand as either online or offline demands. We assume a network flow model which is a good approximation of power flows in the power grids in certain high-voltage operation regimes. To the best of our knowledge, there is no prior study of energy storage networks in an online setting which provides provable guarantees.

Our work is closely related to the classical max-flow and multicommodity flow problems and their generalized flow variants with decay (e.g., [40]) and online variants (e.g., [2, 3, 12]) in terms of the model and techniques. However, these problems have not been studied in combination with storage units subject to a decay factor in an online setting. Our online primal dual approach in the case of public storage units is similar to Buchbinder and Naor [12] and Devanur and Jain [18] and more recent work on online covering and packing problems with convex objectives in [14, 11, 4, 22]. Furthermore, our work is related to inventory problems such as the multi-item lot sizing and joint replenishment problems in the planning aspect. In these problems, one wants to balance the cost of surplus inventory maintenance for future demand against the cost of frequent inventory replenishment; both offline and online settings have been well-studied [26, 31, 7, 10].

Other Related Work

Competitive analysis has been applied in other power management problems. Lu et al. [30] studied the problem of generation scheduling for micro-grids with renewable power sources. Chau et al. [15] designed an online algorithm for single storage operation with uncertain prices and renewable generation. Decay factors have been studied previously in different contexts. Babaioff et al. [5] investigated a variant of the classical secretary problem with discounts. More broadly speaking, deterioration, perishability and lifetime constraints of goods have been studied in numerous mathematical models and optimization problems [33]. In addition to the storage challenges of renewable energy, the economic issues around the energy market and pricing have been investigated by various communities (e.g., [24, 9]).

Our Contributions

In this paper, we develop algorithmic techniques for the *Online Energy Storage Management Problem (OESMP)*. In addition to bridging between the smart grid and TCS communities, we believe our techniques can be of its own interest in analyzing other routing and storage management problems.

Given a network represented by a directed graph $G = (V, E)$ with n nodes and m capacitated edges, we want to route power from online supplies to demands, online or offline, over T time steps using storage units subject to decay factor γ. We consider two main storage unit settings: 1) private storage units which are dedicated storage units co-located with power supplies, and 2) public storage units which are networked units that can store energy from any power supplies with a routing path. Demands have utility functions W and supplies have either zero production cost in the case of renewable power supplies or convex production cost, of the form $Q(z) = az^{\rho}$, in the case of traditional power supplies.

We assume a network flow model which is equivalent to the widely adopted "DC approximation" of AC power flow for power transmission network [38] when power flow routing is possible. Power flow routing is possible if the flows on all edges can be directly controlled as long as conservation and capacity constraints are satisfied. This holds naturally when the network topology is a tree, which is the case for some bulk transmission networks [16] and is approximately true for transmission networks in which *congestion*[1] does not involve cycles [13]. For transmission networks with general network topologies, this may be enabled by Flexible AC Transmission system (FACTS) devices such as phase shifters or smart wires [25]. The flow model can also be used to approximate power flow on distribution networks when line capacity constraints are considered (cf. simplified DistFlow model [6]). More details are provided in Appendix A.

In OESMP, our goal is to route power from supplies to demands and maximize the total utility of satisfied demands less the total production cost of routed power over a finite time horizon of T. We design online deterministic algorithms with poly-logarithmic[2] competitive ratios against the optimal offline algorithm. For simplicity, we assume a single decay factor γ for storage units and a single power exponent ρ for convex cost functions in the following theorem statements. Furthermore, we assume the marginal utility W' is bounded between $w_{\max}$ and $w_{\min} > 0$ and let $\Delta_w = w_{\max}/w_{\min}$. Note n (m) is the number of nodes (edges).

In the *private storage unit setting* with online demands (Section 3), we consider only renewable sources and maximize the utility of satisfied demands. First, we show that an intuitive greedy algorithm achieves a constant competitive ratio if the utility functions are linear and identical.

▶ **Theorem 1.** *For OESMP with private storage units and online demands, there exists a deterministic online algorithm with the constant competitive ratio of* 3 *in the case of uniform utilities.*

We analyze the greedy algorithm using a dual-fitting approach. We show that the congestion on links and storage units leads to a natural dual construction corresponding to the flow in each time step. However, the decay factor introduces strong dependency between different time steps which requires a global evaluation of the dual vectors across the time. We next extend our result to obtain an algorithm with a logarithmic competitive ratio for the private storage unit setting with concave utility functions.

▶ **Theorem 2.** *For OESMP with private storage units and online demands, there exists a deterministic online algorithm with the competitive ratio of* $O(\log \Delta_w)$ *in the case of concave utilities.*

[1] In power systems, thermal constraints limit the amount of power that can be routed through a transmission line. If the maximum is reached for a line, we say the line is congested. Similarly, we say a path of multiple consecutive lines is congested if at least one of the lines on the path is congested.

[2] In this paper, we say a factor is *poly-logarithmic* if it is poly-logarithmic with respect to system parameters n, Δ_w and γ; and not necessarily with respect to the input size.

In the *public storage unit setting* (Section 4), we consider both traditional and renewable sources and the storage units that can be used by any sources. We show that online demands are hard to cope with:

▶ **Theorem 3.** *For OESMP with public storage units and online demands, the competitive ratio of any online algorithm is $\Omega(n)$ even in the case of uniform utilities.*

Therefore, for the public storage unit setting, we focus on the offline demand variant. We apply the online primal dual method to a convex program formulation of the problem using Fenchel duality. Our analysis requires connections between the convex production cost functions and their convex conjugates and utilizes similar critical ideas to those recently developed in [18, 14, 11, 4, 22].

We show poly-logarithmic competitive online algorithms when demands are offline. Inspired by the daily power markets, we assume that T is a small constant[3].

▶ **Theorem 4.** *For OESMP with public storage units and offline demands, there exists a deterministic online algorithm with the competitive ratio of $O(\log n + \log \gamma^{-T} + \log \Delta_w)$ in the case of concave utilities.*

▶ **Theorem 5.** *For OESMP with public storage units and offline demands, there exists a deterministic online algorithm with the competitive ratio of $O\left(\rho^{\rho/(\rho-1)}(\log n + \log \gamma^{-T} + \log \Delta_w)\right)$ for $\rho > \rho_0$ in the case of concave utilities and convex costs of the form $Q(z) = az^\rho$, where ρ_0 is some constant arbitrarily close to 1.*[4]

Finally for the general network with cycles where the network flow model does not apply directly, we show a strong lower bound (Appendix B):

▶ **Theorem 6.** *For OESMP with public storage units in the general power network model and offline demands, the competitive ratio of any online algorithm is $\Omega(n^{1/5})$ even in the case of uniform utilities. This holds even if all the links are physically identical.*

2 Notations and Preliminaries

We define the *Online Energy Storage Management Problem (OESMP)* as follows.

Storage Management Problem

We consider a power transmission network with nodes and edges over a finite time horizon T.[5] The network is represented by a directed graph $G = (V, E)$ with n nodes and m edges.[6] Following the DC-approximation framework for high-voltage regimes, we have edge capacities $C : E \to \mathbb{R}^+$ modeling the thermal constraints on transmission lines. A node can be a supply or demand node at different times and some nodes have storage units.

We denote the set of vertices with nonzero power supply (demand) at time step i by S^i (D^i). Supply sets S^i are given online. Each supply node s has a convex production cost

[3] For many markets, $T = 24$ and $\gamma \in [0.9, 1]$. Hence, note $\log \gamma^{-T} < 1.1$ in practice.

[4] We can remove the condition $\rho > \rho_0$ completely and obtain a poly-logarithmic competitive ratio in terms of m, the number of edges, instead of n. The condition arises from our technical analysis and the constant ρ_0 is accompanied by a correspondingly large constant in the big O notation.

[5] For our applications, T is a small constant, e.g., $T = 24$ for 24 hours.

[6] For a directed edge (u, v), we assume power can only move from u to v on this edge. Forcing direction on the edges makes the model only stronger since a bidirectional link between nodes u and v can be simulated in this model by putting two directed edges (u, v) and (v, u).

function $Q_s^i(z)$ for generating z units of power. If s is a renewable power supply, then it has zero production cost up to the generation capacity at that time. If s is a traditional power supply, then $Q_s(z) = a_s z^{\rho_s}$ for some constants a_s and ρ_s, omitting the superscript i.[7] We would need traditional power generators when the renewable ones alone are not enough.

Demand sets D^i may be given online or offline. We assume that S^i and D^i are disjoint after locally satisfying the demand when there also exists some supply at the same node. Each demand node d has a concave utility function W_d (positive and non-decreasing) such that it would receive $W_d(y)$ units of utility for y units of power received. Since W_d is concave, we have (weakly) diminishing returns on each additional unit of power routed. We assume the slope of the utility function is bounded, i.e., $\frac{dW_d(y)}{dy} \in [w_{\min}, w_{\max}]$ where $w_{\min} > 0$ for $i \in [T]$ and $d \in D^i$. Let $\Delta_w := w_{\max}/w_{\min}$.[8]

We denote the set of nodes with storage units by R. Each storage node has a maximum capacity, a decay factor, charging/discharging inefficiency factors, and ramping constraints. A storage node r can store at most L_r units of power across a time step and maintains γ_r fraction of power stored in the process. When charging or discharging, we are subject to inefficiency factors and ramping constraints in that we lose some fraction of the power in these operations and are limited to some maximum amount of charge/discharge rate per time step. Without loss of generality, we focus on the maximum capacity L_r and decay factor γ_r as we can treat the inefficiency factors and ramping constraints similarly.[9]

Our goal is to route power from supplies to demands and maximize the total utility of satisfied demands less the total production cost of routed power. We model power flows using the standard network flow model, which is equivalent to the widely adopted DC approximation to AC power flow when power routing is possible (see Appendix A).

For analysis, we may represent S^i and D^i with additional nodes and edges. For example, for a renewable supply s with $Q_s(z) = 0$ for $z \leq \tau$ and ∞ otherwise, we create a new node s' with zero production cost and connect it to s with an edge of capacity τ. Similarly, for a demand d with, say, a linear utility function up until a threshold τ, we create d' and connect it to d with an edge of capacity τ. We treat supplies and demands on the same node across different time steps separately as independent supplies and demands.

Private and Public Storage Units

We consider two main storage unit settings. In the private storage setting, we have dedicated storage units co-located with power supplies such that each supply node can use only its own storage unit, if it has one[10]. In this setting, only renewable sources are considered. In the public storage setting, we have networked storage units that can store energy from any

[7] For many real-life applications, we model $Q_s(z) = a_s z^2 + b_s z$ for $a_s, b_s > 0$ when considering just the traditional power generators.

[8] In real-life applications, Δ_w describes the difference between the marginal cost of generation for traditional sources over time, and $\log \Delta_w$ is often a small constant in the scope of power management problems.

[9] Suppose a storage node r is subject to charging/discharging inefficiency factors (γ_+/γ_-) and ramping constraints (τ_+/τ_-). In our model, we would add a new node r' and the following edges: an edge (r, r') with capacity τ_+ and decay factor γ_+, and an edge (r', r) with capacity τ_- and decay factor γ_-. Node r becomes an ordinary connection node and node r' becomes a storage node with the same operation characteristics as the original r.

[10] This corresponds to the important practical case where the storage is co-located with some renewable generation source and is used to smooth the output of random renewable generation [8]. Given the physical size of the inverter and other equipments in such settings, it is natural to assume that power always flows from the generation sites (with both renewable generation and storage) to the grid but not in the other direction.

power supplies (including traditional sources) with a routing path. Consequently, a unit of power can be stored on multiple storage units over time.

Competitive Analysis

In the online paradigm, supply set S^i arrives at each time step i. Demand set D^i may be given online or offline depending on the storage unit setting. Upon the arrival of S^i, we need to dispatch power from supplies in S^i and storage units to demands in D^i, and possibly to other storage units. To evaluate the performance of online algorithms, we use the standard competitive analysis framework. An *online* algorithm is *c-competitive* if its achieved objective value ALG is at least $\frac{1}{c}$ fraction of the optimal *offline* algorithm's value OPT (which knows the entire input in advance) up to an additive constant in all problem instances, i.e., there exist c and a constant c_0 such that $\mathsf{ALG} \geq \frac{1}{c}\mathsf{OPT} - c_0$.

3 Private Storage Units

In this section, we consider the private storage setting in which the storage units used by a supply node are located at that same node. Our goal is to maximize the total utility of satisfied demands using only the renewable power supplies. As discussed in Section 2, we assume, without loss of generality, that the renewable supplies S^i have unbounded zero-cost power generation.

We show strong competitive guarantees for the private storage setting given that both the supply nodes S^i and the demand nodes D^i are arriving online. In Section 3.1, we analyze an intuitive greedy algorithm in the simplified case with uniform utility functions where all demands have the same linear utility function $W(y) = y$ up to the limit $y = 1$, i.e., each demand requires at most 1 unit of power. We show that the greedy algorithm has a constant competitive ratio against the optimal offline algorithm which knows all the supplies and demands in advance.

In contrast to the uniform utility case, we can show that the greedy algorithm is not competitive when the utility functions are arbitrary concave functions. In Section 3.2, we show that we can still design a competitive algorithm for the concave utility case using the constant competitive greedy algorithm as a black-box. More precisely, we show that an algorithm with the competitive ratio c for the uniform utility case can be modified to obtain an algorithm with the competitive ratio of $O(c \cdot \log \Delta_w)$ for the general concave utility case.

3.1 Uniform Utilities

We consider the following primal-dual linear program formulation of the problem with private storage units. Assume that every supply node $s \in S^i$ is connected to a private storage unit r^s. For time step i and demand node $d \in D^i$, let $P^+(i, d)$ denote the set of all paths ending at d and starting from a supply node or a storage node. For time step i and storage node r, let $p^+(i, r)$ denote the path ending at r and starting from the supply s which owns the storage node. Let $P^-(i, r)$ denote the set of paths from r to a demand node in D^i. Note that since the storage units are co-located with supplies, the paths $p^+(i, r)$ are edge-disjoint. We define $P^+_D = \bigcup_{i,d\in D^i} P^+(i, d)$, $P^+ = P^+_D \cup \bigcup_{i,r}\{p^+(i, r)\}$, and $P^- = \bigcup_{i,r} P^-(i, r)$.

For a path p, the variable x_p denotes the amount of flow passing through the path. Furthermore, for a path $p \in P^+$ the parameters $i(p)$ and $v(p)$ denote the time step in which p lies and the node at the beginning of p. For a storage node r and a time step i, y^i_r denotes the amount of power stored in the node r at the *end* of time step i.

The first set of primal constraints ensure that the total flow routing through an edge does not violate the edge capacity. The second set of constraints bounds the amount of energy in storage units. The last set of primal constraints ensures the feasibility of flow over time: we may assume that a storage node sends the stored energy to the next time step and receives energy from the previous time step. Intuitively, the last constraint implies that the total outgoing flow cannot be more than the total incoming flow.

$$\begin{array}{llr}
\max & \sum_{p \in P_D^+} x_p & (\mathbb{P}) \\
\forall e & \sum_{p \ni e} x_p \le C_e & (\alpha_e) \\
\forall i, r & y_r^i \le L_r & (\beta_r^i) \\
\forall i, r & y_r^i + \sum_{p \in P^-(i,r)} x_p \le \gamma_r y_r^{i-1} + x_{p^+(i,r)} & (\tau_r^i) \\
 & y_r^0 = 0; \quad x_p, y_r^i \ge 0 &
\end{array}
\qquad
\begin{array}{llr}
\min & \sum_e C_e \alpha_e + \sum_{i,r} L_r \beta_r^i & (\mathbb{D}) \\
\forall p \in P_D^+ & \sum_{e \in p} \alpha_e + \tau_{v(p)}^{i(p)} \ge 1 & (\mathrm{D1}) \\
\forall i, r & \sum_{e \in p^+(i,r)} \alpha_e \ge \tau_r^i & (\mathrm{D2}) \\
\forall i, r & \beta_r^i + \tau_r^i \ge \gamma_r \tau_r^{i+1} & (\mathrm{D3}) \\
 & \alpha, \beta, \tau \ge 0; \forall v \in S^i : \tau_{i,v} = 0 &
\end{array}$$

Greedy Algorithm

Upon the arrival of S^i and D^i, we route the maximum power flow from S^i and the storage units to the demand nodes in D^i. We store the residual power generated at a supply in its private storage unit, up to the storage capacity and edge capacities on the path from the supply to the storage. Recall that based on the simplifications in our model, there can be two edges on the path from a supply to its dedicated storage unit.

Dual Construction

We now use a dual-fitting argument to show that the greedy algorithm is indeed constant competitive. For a time step i, let $P^+(i) := \bigcup_{d \in D^i} P^+(i,d)$. After the run of the greedy algorithm, consider the flow paths $\hat{x}$ and the storage amounts $\hat{y}$ chosen by the greedy algorithm, corresponding to the primal solution. Let $\hat{x}^i$ denote the selected flow at time step i. Furthermore, let $\hat{x}(D^i)$ denote the selected flow that satisfies the demands in D^i. We note that by definition, $\hat{x}(D^i)$ is a sub-flow of $\hat{x}^i$. In what follows, we say a flow f on a graph separates a vertex u from v, if f cannot be augmented by a u–v flow in the residual network. We now describe the construction of our dual solution. We then show the feasibility of the dual solution and the bound on its objective.

DC Part I. Let R_ϕ^i denote the set of storage nodes that are *not* separated from D^i by $\hat{x}(D^i)$. We choose a minimum cut separating $S^i \cup (R \setminus R_\phi^i)$ from D^i. For every edge e in this cut, we set $\alpha_e = 1$.

DC Part II. We repeat the following process for every j by starting from $j = T$ and ending at $j = 1$: For every $r \in R_\phi^j$, set $\tau_r^j = 1$. Let i denote the last step before j for which $\hat{y}_r^i = L_r$ in the greedy solution. If no such i exists, set $i = 0$. For $k = \{j-1, j-2, \ldots, i+2, i+1\}$, set the dual variable $\tau_r^k = \gamma_r \tau_r^{k+1}$. Furthermore, if $i > 0$ set $\beta_r^i = \gamma_r \tau_r^{i+1}$. We note that in this process, we may reassign values multiple times to τ_r^i for some i and r[11]. Hence, it is important to iterate j from T to 1.

[11] In fact, the assigned values can only be non-decreasing.

DC Part III. For every $i \in [T]$ and storage node r, if the path $p^+(i,r)$ is congested in $\hat{x}^i$, we choose a congested edge $e \in p^+(i,r)$. If α_e is not set in Part I, we set $\alpha_e = \tau_r^i$.

We now need to prove the feasibility of the constructed dual vector. See Appendix C for proofs.

▶ **Lemma 7.** *The dual vector $\langle \alpha, \beta, \tau \rangle$ constructed by DC-Parts I-III is feasible.*

It now remains to prove a bound on the dual objective. Let GA denote the total primal value of the greedy solution.

Proof of Theorem 1. We analyze the cost we incur at every part of the dual construction separately. We show that the dual cost in each part can be upper bounded by GA.

For a flow F, let $|F|$ denote the amount of flow routed by F. Recall that for a flow F on a graph, if there is no augmenting path from a set U to a set V (i.e., F separates U from V), then the size of the minimum cut separating U and V is upper bounded by $|F|$. Now consider $\hat{x}(D^i)$ for some i. By definition, $R \setminus R^i_\phi$ is separated by $\hat{x}(D^i)$ from D^i. Furthermore, since the greedy algorithm chooses a maximum flow, we know that $\hat{x}^i(D^i)$ cannot be augmented by a path from a supply node in S^i to a demand node. Therefore $\hat{x}(D^i)$ is separating $S^i \cup (R \setminus R^i_\phi)$ from D^i. The total dual cost of the minimum cut we choose in Part I for some i is upper bounded by $|\hat{x}^i(D^i)|$. Hence, the total cost we incur in Part I is bounded by $\sum_i |\hat{x}^i(D^i)| \le \text{GA}$.

The cost we incur in Part II is $\sum_{i,r} L_r \beta_r^i$. The variable β_r^i is positive only if r is full at time i, i.e., $\hat{y}_r^i = L_r$. By Lemma 14, we know that the stored power will be used in at most $\log_{\gamma_r}(\beta_r^i)$ steps. Therefore the primal gain from the power stored at time step i in r is at least $\gamma_r^{\log_{\gamma_r}(\beta_r^i)} \hat{y}_r^i = L_r \beta_r^i$. Furthermore, suppose for two time steps $i_2 > i_1$, we have that $\beta_r^{i_1} > 0$ and $\beta_r^{i_2} > 0$. By Part II of the construction, we know that r should become empty for some time step $j \in \{i_1 + 1, \dots, i_2 - 1\}$. Thus the power stored in r in time steps i for which $\beta_r^i > 0$, are disjoint. Therefore now we can charge the cost $\sum_{i,r} L_r \beta_r^i$ to the utility we gain from dispatching power stored in the storage units at time steps in which $\beta_r^i > 0$.

The cost we incur in Part III can be upper bounded as follows. Suppose $\tau_r^i > 0$ for some i and r. The cost we incur is $\tau_r^i C_e$ for some congested edge $e \in p^+(i,r)$. We note that the flow passing through e is either satisfying a demand in D^i, or it is being stored in r. In the former, the utility we gain is 1 per unit of power. In the latter, by Lemma 14, we gain utility at rate at least $\gamma_r^{\log_{\gamma_r}(\tau_r^i)} = \tau_r^i$. Therefore our primal gain from the flow routed on e is at least $C_e \tau_r^i$. Now since the storage nodes are co-located, all the paths $p^+(i,r)$ are disjoint. Therefore we can bound the total cost incurred in Part III by GA.

Finally summing over the three parts, we have that the dual objective is at most $3\,\text{GA}$. The theorem follows from Lemma 7 and weak duality. ◀

3.2 Concave Utilities

In this section, we demonstrate a simple reduction from the variant with concave utility functions to the variant with uniform linear functions losing only a logarithmic factor. We then use the algorithm of Theorem 1 as a blackbox to solve the case of concave utilities and obtain the competitive ratio of $O(\log \Delta_w)$.

Proof Sketch of Theorem 2. The main idea is to reduce an instance of the problem with concave utility functions to $O(\log \Delta_w)$ instances of the problem with uniform utility functions. Since $W_d{}'(x) \in [w_{\min}, w_{\max}]$ for each demand node d, we can construct $O(\log \Delta_w)$ instances where each ${W_d}'$ is approximately constant within each instance. Then, we solve each instance independently using a constant competitive algorithm. Due to the space constraints, we defer the complete proof to Appendix C. ◀

4 Public Storage Units

We consider the general setting with public storage units on the network. A supply node can access any storage unit as long as there is a path, and, consequently, a unit of power can be stored on multiple storage units over time. As Theorem 3 shows online demands are hard to satisfy (see Appendix D for proof), we focus on the offline demands in this section. Note offline demands naturally model scenarios where consumer demands', say, hourly variation is predictable.

We investigate two specific cases: the case of renewable power generation in Section 4.1 and the case of combined traditional and renewable power generation in Section 4.2. In the first case, we want to route power from renewable supplies to demands assuming the production cost is zero. In the second case, we still route power but the supplies are equipped with both traditional and renewable power generators and have time-varying convex production costs. Supplies are arriving online while demands are given offline and have concave utility functions.

We design poly-logarithmic competitive online algorithms using the online primal dual method on convex programming formulations. Our approach is closely related to Buchbinder and Naor [12] and Devanur and Jain [18] and more recent work on online covering and packing problems with convex objectives in [14, 11, 4, 22]. For analysis, we use the following bicriteria notion of competitive algorithms: An algorithm is (c_1, c_2)-*competitive* if it routes the total flow of amount at least $\frac{1}{c_1}$ of the optimal and the load on each edge is at most c_2, where the *load* of an edge is the ratio of the total flow going through it divided by its capacity. Ideally, the total bandwidth allocated for flows should not exceed the capacity.

Time-Expanded Graph

We use *time-expanded graph* $G^* = (V^*, E^*)$ constructed as follows. For $i = 1, \dots, T$, we create a time-copy of G, $G^i = (V^i, E^i)$. To represent storage, we create *storage edges* between time-copies of G; for each node v and time step i, we create an edge of capacity L_v from v^i to v^{i+1}. Let $S^* = \bigcup_i S^i$ and $D^* = \bigcup_i D^i$. Instead of creating a node for each individual demand as in Section 2, we add a single node d^* as the designated super-demand. For each $i \in [T]$ and demand $d \in D^i$, we add an edge of infinite capacity from d^i to d^*; these are *demand edges* and we use D^* to denote both the demand edges and corresponding demands.

For $s \in S^*$ and $d \in D^*$, let $P(s, d)$ be the set of simple paths in G^* from supply s to demand d. Let $P(s, \cdot) = \bigcup_d P(s, d)$, $P(\cdot, d) = \bigcup_s P(s, d)$, and P be the set of all simple paths from supplies to demands. For a routing path p, we denote the corresponding supply and demand nodes by $s(p)$ and $d(p)$, respectively; we simply use s and d if p is clear from the context. For simplicity, we assume a single decay factor γ for all storage units. We define:

$\gamma(p) :=$ overall decay due to storage edges on p;

$\gamma(p, e) :=$ overall decay due to storage edges on p preceeding edge $e \in p$.

In our case, $\gamma(p) = \gamma^{(\text{number of storage edges on } p)}$ and $\gamma(p, e) = \gamma^{(\text{number of storage edges on } p \text{ before } e)}$. Let $l_{\max} = nT$ be the maximum routing path length, and $\gamma_{\min} = \gamma^T$ be the greatest overall decay.

Algorithm 1 Online Algorithm for Concave Utilities

```
1: Let $y_d = \sum_{p \in P(\cdot,d)} \gamma(p) x_p, \forall d.$                    ▷ $y$ determined in terms of $x$
2: for $i = 1, \ldots, T$ do
3:     for $s \in S^i$ in arbitrary order do
4:         while $P' = \{p \in P(s,\cdot) : \sum_{e \in p} \gamma(p,e)\alpha_e < \gamma(p) {W_{d(p)}}'(\lambda_{d(p)})\}$ is not empty do
5:             Update continuously:
6:                 $p = \arg\max_{p \in P'} \gamma(p) {W_{d(p)}}'(\lambda_{d(p)})$
7:                 $dx_p = 1$                                    ▷ Increase $x_p$ at a uniform rate
8:                 $\frac{d\lambda_{d(p)}}{dx_p} = \gamma(p)$
9:                 $\frac{d\alpha_e}{dx_p} = c\left(\frac{\gamma(p,e)\alpha_e}{C_e} + \frac{\gamma(p){W_{d(p)}}'(\lambda_{d(p)})}{l_{\max} C_e}\right), \forall e \in p$      ▷ $c \geq 1$ is some parameter
10:        end while
11:    end for
12: end for
```

4.1 Concave Utilities

In this section, we consider only renewable power generation with zero production cost and route power from supplies to demands. We model the demand nodes' utility functions with monotonically nondecreasing concave functions W_d such that demand node d gains the utility of $W_d(f)$ for receiving f units of flow. There is diminishing returns on each additional flow routed.

We consider the following primal-dual convex program formulation. For each path p, x_p denotes the amount of flow passing through the path. For demand d, y_d denotes the total amount of flow routed to the demand and is set to equal $y_d = \sum_{p \in P(\cdot,d)} \gamma(p) x_p$. W_d is a monotonically nondecreasing concave function and we define $\widehat{W_d}(\lambda) := W_d(\lambda) - \lambda {W_d}'(\lambda)$.

$$\begin{array}{llcll}
\max & \sum_d W_d(y_d) & \quad & \min & \sum_e C_e \alpha_e + \sum_d \widehat{W_d}(\lambda_d) \\
\forall e & \sum_{p \ni e} \gamma(p,e) x_p \leq C_e & & \forall p \in P & \sum_{e \in p} \gamma(p,e)\alpha_e \geq \gamma(p) {W_{d(p)}}'(\lambda_{d(p)}) \\
\forall d \in D^* & y_d \leq \sum_{p \in P(\cdot,d)} \gamma(p) x_p & & & \alpha, \beta, \lambda \geq 0 \\
 & x, y \geq 0 & & &
\end{array}$$

We first prove Algorithm 1 is bicriteria competitive.

▶ **Lemma 8.** *For any $c \geq 1$, Algorithm 1 is $\left(2c+1, \frac{O(\log n + \log \gamma^{-T} + \log \Delta_w)}{c}\right)$-competitive.*

Proof. We show that Algorithm 1 returns a primal solution that violates the edge capacity constraints by some factor and a feasible dual solution. From the ratio of the primal and dual objective values, $\mathcal{P}$ and $\mathcal{D}$, we obtain the stated competitive ratio.

For a supply s, P' in Line 4 is nonempty if there is a violated dual constraint. As long as P' is nonempty, we continuously increase x_p and dual variables correspondingly. Since all variables increase monotonically and the first derivatives ${W_d}'$ are monotonically non-increasing, violated dual constraints for $p \in P(s,\cdot)$ eventually become satisfied and no previously satisfied constraints become violated. Hence, the while loop terminates with no violated dual constraints for $p \in P(s,\cdot)$, and the algorithm terminates with a feasible dual solution.

The lemma follows from the following claims (proved in Appendix D) and the weak duality:

▶ **Claim 9.** *The load on each edge is at most* $\frac{O(\log n + \log \gamma^{-T} + \log \Delta_w)}{c}$ *in the primal solution.*

▶ **Claim 10.** *At termination,* $\mathcal{D} \leq (2c+1) \cdot \mathcal{P}$.

◀

We can now prove Theorem 4. For checking the condition and choosing path p in Lines 4–6 in Algorithm 1, we can use a backward variant of Dijkstra's algorithm in polynomial time (details in Appendix D):

Proof of Theorem 4. We choose $c = c' \cdot (\log n + \log \gamma^{-T} + \log \Delta_w)$ for the constant c' that results from the analysis in Lemma 8. Then, we get an $\big(O(\log n + \log \gamma^{-T} + \log \Delta_w), 1\big)$-competitive algorithm. ◀

4.2 Concave Utilities and Convex Costs

We consider the more general case where each supply is equipped with both renewable and traditional power generators. Each supply node s individually manages its own renewable and traditional power generation and only pays cost for using the traditional generators. It generates power according to the production cost function $Q_s(z) = a_s z^{\rho_s}$ which changes from time to time depending on the renewable power generation. The production cost functions are given online.

We consider the following primal-dual convex program formulation. For each path p, x_p denotes the amount of flow passing through the path. For demand d, y_d denotes the total amount of flow routed to the demand and is set to equal $y_d = \sum_{p \in P(\cdot,d)} \gamma(p) x_p$. For supply s, z_s denotes the total power generated using the traditional power generators and is set to equal $z_s = \sum_{p \in P(s,\cdot)} x_p$. We define $\widehat{W_d}(\lambda) := W_d(\lambda) - \lambda {W_d}'(\lambda)$ and Q^* is the convex conjugate of Q defined to be $Q^*(\mu) := \sup_{z \geq 0}\{\mu z - Q(z)\}$.

$$\begin{array}{rlrl}
\max & \sum_d W_d(y_d) - \sum_s Q_s(z_s) & \min & \sum_e C_e \alpha_e + \sum_d \widehat{W_d}(\lambda_d) + \sum_s Q_s^*(\mu_s) \\
\forall e \in E & \sum_{p \ni e} \gamma(p,e) x_p \leq C_e & \forall p \in P & \sum_{e \in p} \gamma(p,e)\alpha_e + \mu_{s(p)} \geq \gamma(p) {W_{d(p)}}'(\lambda_{d(p)}) \\
\forall d \in D^* & \sum_{p \in P(\cdot,d)} \gamma(p) x_p \geq y_d & & \alpha, \lambda, \mu \geq 0 \\
\forall s \in S^* & \sum_{p \in P(s,\cdot)} x_p \leq z_s & & \\
 & x, y, z \geq 0 & &
\end{array}$$

We show that Algorithm 2, given in Appendix D, is poly-logarithmic competitive. For concreteness, we prove Theorem 5 with $\rho_0 = 1.44$. The constant ρ_0 can be replaced with a smaller constant arbitrarily close 1 and with a correspondingly large multiplicative constant in the competitive ratio. We first prove the following lemma:

▶ **Lemma 11.** *For any* $c \geq 1$, *Algorithm 2 (in Appendix D) is* $\left((2c+1)\rho^{\rho/(\rho-1)} + 1, \frac{O(\log n + \log \gamma^{-T} + \log \Delta_w)}{c}\right)$*-competitive where* $\rho = \max_s \rho_s > 1.44$.

Proof. The proof is similar to that of Lemma 8. We need to prove claims about the load and competitive ratio. Analyzing the competitive ratio is more difficult because of the convex cost functions and requires a different approach.

The lemma follows from the following claims (with complete proofs in Appendix D) and the weak duality.

▶ **Claim 12.** *The load on each edge is at most* $\frac{O(\log n+\log\gamma^{-T}+\log\Delta_w)}{c}$ *in the primal solution.*

▶ **Claim 13.** *At termination,* $\mathcal{D}\le\left((2c+1)\rho^{\rho/(\rho-1)}+1\right)\cdot\mathcal{P}$.

Proof Sketch of Claim 13. Let $\mathcal{D}_0=\sum_e C_e\alpha_e+\sum_s Q_s^*(\mu_s)$; so, $\mathcal{D}=\mathcal{D}_0+\sum_d\widehat{W}_d(\lambda_d)$. Assume $\mathcal{D}_0\le(2c+1)\rho^{\rho/(\rho-1)}\mathcal{P}$. Since $\widehat{W}_d(z)\le W_d(z),\forall z\ge 0$, it would follow that $\sum_d\widehat{W}_d(\lambda_d)\le\sum_d W_d(y_d)\le\mathcal{P}$. Then, $\mathcal{D}=\sum_d\widehat{W}_d(\lambda_d)+\mathcal{D}_0\le\left((2c+1)\rho^{\rho/(\rho-1)}+1\right)\mathcal{P}$, and the claim would follow.

To show $\mathcal{D}_0\le(2c+1)\rho^{\rho/(\rho-1)}\mathcal{P}$, we prove $d\mathcal{P}\ge\frac{1}{\sigma}d\mathcal{D}_0$ for $\sigma=(2c+1)\rho^{\rho/(\rho-1)}$ and $\rho>1.44$ when we route power. This reduces to showing

$$\left(1-\frac{2c}{\sigma}\right)\mu_s-{Q_s}'(z_s)\ge\frac{1}{\sigma}(Q_s^*)'(\mu_s)\frac{d\mu_s}{dz_s}\ ,$$

which is satisfied by our choice of updates to primal and dual variables. For a ρ_0 constant smaller than 1.44, we would need to have a multiplicative factor greater than $2c+1$. Due to the space constraints, we defer the complete proof to Appendix D. ◀

◀

We can now prove Theorem 5. For the path construction routine in Algorithm 2, we find the routing paths in the same manner as in Section 4.1:

Proof of Theorem 5. Let $\rho=\max_s\rho_s$. We choose $c=c'\cdot(\log n+\log\gamma^{-T}+\log\Delta_w)$ for the constant c' that results from the analysis in Lemma 11. Then, we get an $\left(O\left(\rho^{\rho/(\rho-1)}(\log n+\log\gamma^{-T}+\log\Delta_w)\right),1\right)$-competitive algorithm. ◀

References

1 Rajeev Alur, Sampath Kannan, Kevin Tian, and Yifei Yuan. On the complexity of shortest path problems on discounted cost graphs. In *Proceedings of the 7th International Conference on Language and Automata Theory and Applications*, LATA'13, pages 44–55, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-37064-9_6.

2 James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 44(3):486–504, May 1997. doi:10.1145/258128.258201.

3 B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive online routing. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'93, pages 32–40, 1993.

4 Yossi Azar, Ilan Reuven Cohen, and Debmalya Panigrahi. Online covering with convex objectives and applications. *arXiv:1412.3507*, Dec 2014. URL: http://arxiv.org/abs/1412.3507.

5 Moshe Babaioff, Michael Dinitz, Anupam Gupta, Nicole Immorlica, and Kunal Talwar. Secretary problems: Weights and discounts. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'09, pages 1245–1254, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1496770.1496905.

6 M.E. Baran and F.F. Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transactions on Power Delivery*, 4(2):1401–1407, apr 1989. doi:10.1109/61.25627.

7 Marcin Bienkowski, Jaroslaw Byrka, Marek Chrobak, Lukasz Jeż, Dorian Nogneng, and Jiří Sgall. Better approximation bounds for the joint replenishment problem. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'14, pages 42–54. SIAM, 2014. URL: http://dl.acm.org/citation.cfm?id=2634074.2634078.

8 E. Bitar, R. Rajagopal, P. Khargonekar, and K. Poolla. The Role of Co-Located Storage for Wind Power Producers in Conventional Electricity Markets. In *Proc. of American Control Conference (ACC)*, pages 3886–3891, 2011.

9 E. Bitar and Yunjian Xu. On incentive compatibility of deadline differentiated pricing for deferrable demand. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 5620–5627, Dec 2013. doi:10.1109/CDC.2013.6760775.

10 N. Buchbinder, T. Kimbrelt, R. Levi, K. Makarychev, and M. Sviridenko. Online make-to-order joint replenishment model: Primal dual competitive algorithms. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'08, pages 952–961, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1347082.1347186.

11 Niv Buchbinder, Shahar Chen, Anupam Gupta, Viswanath Nagarajan, and Joseph (Seffi) Naor. Online convex covering and packing problems. *arXiv:1412.8347*, Dec 2014. URL: http://arxiv.org/abs/1412.8347.

12 Niv Buchbinder and Joseph (Seffi) Naor. Improved bounds for online routing and packing via a primal-dual approach. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'06, pages 293–304, Washington, DC, USA, 2006. IEEE Computer Society. doi:10.1109/FOCS.2006.39.

13 California Independent System Operator. Annual Report on Market Issues and Performance, 2014. http://www.caiso.com/Documents/2014AnnualReport_MarketIssues_Performance.pdf.

14 T-H. Hubert Chan, Zhiyi Huang, and Ning Kang. Online convex covering and packing problems. *arXiv:1502.01802*, Apr 2015. URL: http://arxiv.org/abs/1502.01802.

15 Chi-Kin Chau, Guanglin Zhang, and Minghua Chen. Cost Minimizing Online Algorithms for Energy Storage Management with Worst-case Guarantee. *IEEE Transactions on Smart Grid*, nov 2015. URL: http://arxiv.org/abs/1511.07559, arXiv:1511.07559.

16 In-Koo Cho. Competitive equilibrium in a radial network. *RAND Journal of Economics*, pages 438–460, 2003.

17 Paul Denholm, Erik Ela, Brendan Kirby, and Michael Milligan. The role of energy storage with renewable electricity generation. Technical Report NREL/TP-6A2-47187, National Renewable Energy Laboratory, January 2010.

18 Nikhil R. Devanur and Kamal Jain. Online matching with concave returns. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC'12, pages 137–144, New York, NY, USA, 2012. ACM. doi:10.1145/2213977.2213992.

19 European Commission. Energy Roadmap 2050, 2011. https://ec.europa.eu/energy/en/topics/energy-strategy/2050-energy-strategy.

20 J Duncan Glover, Mulukutla Sarma, and Thomas Overbye. *Power System Analysis & Design, Fifth Edition.* Cengage Learning, 2012.

21 L. Huang, J. Walrand, and K. Ramchandran. Optimal Demand Response with Energy Storage Management. In *Proc. of IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pages 61–66, 2012. doi:10.1109/SmartGridComm.2012.6485960.

22 Zhiyi Huang and Anthony Kim. Welfare maximization with production costs: A primal dual approach. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, pages 59–72. SIAM, 2015. URL: `http://dl.acm.org/citation.cfm?id=2722129.2722135`.

23 Mark Z. Jacobson, Mark A. Delucchi, Guillaume Bazouin, Zack A. F. Bauer, Christa C. Heavey, Emma Fisher, Sean B. Morris, Diniana J. Y. Piekutowski, Taylor A. Vencill, and Tim W. Yeskoo. 100% clean and renewable wind, water, and sunlight (WWS) all-sector energy roadmaps for the 50 United States. *Energy Environ. Sci.*, 8(7):2093–2117, Jul 2015. `doi:10.1039/C5EE01283J`.

24 Thomas Kesselheim, Robert Kleinberg, and Eva Tardos. Smooth online mechanisms: A game-theoretic problem in renewable energy markets. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC'15, pages 203–220, New York, NY, USA, 2015. ACM. `doi:10.1145/2764468.2764487`.

25 Frank Kreikebaum, Debrup Das, Yi Yang, Frank Lambert, and Deepak Divan. Smart Wires — A distributed, low-cost solution for controlling power flows and monitoring transmission lines. In *2010 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT Europe)*, pages 1–8. IEEE, oct 2010. `doi:10.1109/ISGTEUROPE.2010.5638853`.

26 Retsef Levi, Robin O. Roundy, and David B. Shmoys. Primal-dual algorithms for deterministic inventory problems. *Math. Oper. Res.*, 31(2):267–284, February 2006. `doi:10.1287/moor.1050.0178`.

27 David Lindley. The energy storage problem. *Nature*, 463(7), January 2010.

28 S. H. Low. Convex Relaxation of Optimal Power Flow, Part I: Formulations and Equivalence. *ArXiv e-prints*, May 2014. `arXiv:1405.0766`.

29 S. H. Low. Convex Relaxation of Optimal Power Flow, Part II: Exactness. *ArXiv e-prints*, May 2014. `arXiv:1405.0766`.

30 Lian Lu, Jinlong Tu, Chi-Kin Chau, Minghua Chen, and Xiaojun Lin. Online energy generation scheduling for microgrids with intermittent energy sources and co-generation. *ACM SIGMETRICS Performance Evaluation Review*, 41(1):53, jun 2013. URL: `http://dl.acm.org/citation.cfm?id=2494232.2465551`, `doi:10.1145/2494232.2465551`.

31 Viswanath Nagarajan and Cong Shi. Approximation algorithms for inventory problems with submodular or routing costs. *arXiv:1504.06560*, April 2015. URL: `http://arxiv.org/abs/1504.06560`.

32 National Renewable Energy Laboratory. The Value of Energy Storage for Grid Applications, 2013. URL: `http://www.nrel.gov/docs/fy13osti/58465.pdf`.

33 Julia Pahl and Stefan Voß. Integrating deterioration and lifetime constraints in production and supply chain planning: A survey. *European Journal of Operational Research*, 238(3):654–674, 2014. `doi:10.1016/j.ejor.2014.01.060`.

34 J. Qin, R. Sevlian, D. Varodayan, and R. Rajagopal. Optimal Electric Energy Storage Operation. In *Proc. of IEEE Power and Energy Society General Meeting*, pages 1–6, 2012. `doi:10.1109/PESGM.2012.6345242`.

35 J. Qin, H. I. Su, and R. Rajagopal. Storage in Risk Limiting Dispatch: Control and Approximation. In *Proc. of American Control Conference (ACC)*, pages 4202–4208, 2013.

36 Junjie Qin, Yinlam Chow, Jiyan Yang, and Ram Rajagopal. Distributed Online Modified Greedy Algorithm for Networked Storage Operation under Uncertainty. *Smart Grid, IEEE Transactions on*, PP(99):1, jun 2014. URL: `http://arxiv.org/abs/1406.4615`, `arXiv:1406.4615`, `doi:10.1109/TSG.2015.2422780`.

37 SB-350 Clean Energy and Pollution Reduction Act of 2015, 2015. `https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201520160SB350`.

38 Brian Stott, Jorge Jardim, and Ongun Alsaç. Dc power flow revisited. *Power Systems, IEEE Transactions on*, 24(3):1290–1300, 2009.

39 H. I. Su and A. El Gamal. Modeling and Analysis of the Role of Energy Storage for Renewable Integration: Power Balancing. *IEEE Transactions on Power Systems*, 28(4):4109–4117, 2013. doi:10.1109/TPWRS.2013.2266667.

40 Kevin D. Wayne. *Generalized maximum flow algorithms.* PhD thesis, Cornell University, 1999.

41 L. Xie, Y. Gu, A. Eskandari, and M. Ehsani. Fast MPC-Based Coordination of Wind Power and Battery Energy Storage Systems. *Journal of Energy Engineering*, 138(2):43–53, 2012. doi:10.1061/(ASCE)EY.1943-7897.0000071.

A From Electric Power Flow to Network Flow

Bulk electric power grids are operated with Alternating current (AC) power flow, where the physical quantities of interest, such as voltage, current and power, are sinusoidal signals of time. Economic and market operations of the grid are usually solved and scheduled for a slow timescale, such that the controls or set points for the system are modified at a frequency of every 5 minutes or lower. Consequently the sinusoidal signals have stabilized into a steady state for almost all time points in each time slot, and are characterized by constant-frequency sinusoidal waveforms and permit a *phasor representation* [20, Section 2.1]. The AC voltage is then represented as a phasor, denoted by $\mathcal{V}\exp(\mathbf{i}\theta) \in \mathbb{C}$, where $\mathcal{V} \in \mathbb{R}_+$ is the (root-mean-square) voltage magnitude, $\theta \in \mathbb{R}$ is the voltage phase angle, and $\mathbf{i} = \sqrt{-1}$. Provided that current has a similar phasor representation, the resulting power is a complex number, whose real part is referred to as *real power* $\mathcal{P}$ and imaginary part is referred to as *reactive power* $\mathcal{Q}$. Intuitively speaking, real power can be thought of as the actual power that serves the load, while reactive power is a consequence of the phase difference between the current and voltage phasors. Thus the majority of the economic aspect of the grid operation has centered around real power, with reactive power considered often for other purposes such as power quality.

Given an electric grid represented by a graph $G = (V, E)$ consisting of a set V of buses and a set E of lines connecting the buses, the *AC power flow equation* is a set of $2|V|$ nonlinear equations relating the voltage phasors $(\mathcal{V}_v, \theta_v)$ at each bus $v \in V$ and the corresponding complex powers $(\mathcal{P}_v, \mathcal{Q}_v)$, $v \in \mathcal{V}$. Together with additional operational constraints such as line capacity and suitable objectives, one can formulate corresponding optimization problems for the set points of the controllable devices on the grid. In general, such optimization problems are often referred to as *AC optimal power flow* (AC-OPF) problems. Given the nonlinear nature of the AC power flow equations, AC OPF is in general nonconvex and NP-hard [28, 29], so that practical solutions have relied on approximation of the AC power flow equations.

The most widely adopted approximation for bulk electric power networks (transmission networks) is a particular linearization of the AC power flow equations referred to as DC approximation to AC power flow [38]. Assuming that i) the voltage magnitudes over all buses are held constant, ii) all lines are purely inductive (i.e., there is no real power losses due to resistance), and iii) voltage phase differences between buses are small, we can obtain a linear relationship between the nodal real power injections $\mathcal{P}_v$, $v \in V$, and the voltage phase angles θ_v, $v \in V$. In particular, the real power flow through any line $e = (v, u) \in E$ can be written as

$$e \in E, \quad f_e = \mathcal{B}_e(\theta_v - \theta_u), \tag{1}$$

where $\mathcal{B}_e$ is the reciprocal of the reactance of line e. Consequently, by flow conservation on node $v \in V$, we have

$$v \in V, \quad \mathcal{P}_v = \sum_{e=(v,u)} \mathcal{B}_e(\theta_v - \theta_u).$$

Therefore, the differences of bus voltage phase angles θ_v, $v \in V$ determine all the line flows and nodal power injections, and hence fully characterize the operation condition of the system under the DC approximation. As phase angle differences are invariant to a constant shift, we can without loss of generality set the phase angle at a bus v_0, called slack bus, to be 0 and work with the remaining phase angles θ_v, $v \in V\backslash\{v_0\}$. When the flow capacities of lines are considered, together with the fact that the nodal power injection at certain buses may not be controllable, the feasible set of phase angles is then characterized by the constraints

$$v \in V^{\text{Fix}}, \quad g_v - d_v = \sum_{e=(v,u)} \mathcal{B}_e(\theta_v - \theta_u), \tag{2}$$

$$e \in E, \quad \mathcal{B}_e(\theta_v - \theta_u) \leq C_e, \tag{3}$$

where $V^{\text{Fix}} \subset V$ is the set of buses only connected to uncontrollable devices, g_v is the uncontrollable generation at bus v, d_v is the uncontrollable demand at bus v, and C_e is the real power capacity of line e.

To convert the formulation in (2) and (3) in terms of phase angles θ_v, $v \in V$, into a network flow formulation which uses the flows f_e, $e \in E$, as the variables, we observe that, using (1), (2) and (3) can be written as

$$v \in V^{\text{Fix}}, \quad g_v - d_v = \sum_{e=(v,u)} f_e, \tag{4}$$

$$e \in E, \quad f_e \leq C_e. \tag{5}$$

This set of constraints, however, in general is not an equivalent formulation to (2) and (3) as not every collection of flows f_e, $e \in E$ that satisfies these equations will induce a feasible collection of phase angles θ_v, $v \in V$. In particular, when $V^{\text{Fix}} = \emptyset$, we know that the set of feasible θ_v, $v \in V$ lives in an affine subspace of dimension $|V| - 1$. However for general graph with $|E| > |V| - 1$, the set of feasible flows is of dimension $|E|$. Consequently the mapping between f_e, $e \in E$ and θ_v, $v \in V$, defined by (1) is not one-to-one. This mapping would indeed be one-to-one if the graph G is a tree. In this case, it is easy to check that (4) and (5), which is the standard *network flow* constraints, equivalently characterize the set of feasible operation conditions of the system under the DC approximation.

B Hardness of Networks with Cycles

In this section, we show a hardness result for general instances of OESMP when congested transmission lines can form a cycle in the network. We assume the general network flow model described in Appendix A and the network flow formulation given by (4)-(5) where C_e is the capacity and $\mathcal{B}_e$ is the susceptance, the reciprocal of the reactance, of edge e.

Proof of Theorem 6. We prove the lower bound by constructing a hard example. We first consider the network in Figure 1 and then construct a related network with identical edges with the same susceptance and capacity. In the network in Figure 1, the edges are bidirectional and have the same susceptance of (sufficiently large) $\mathcal{B}$ but different capacities, 1 or M, as indicated.

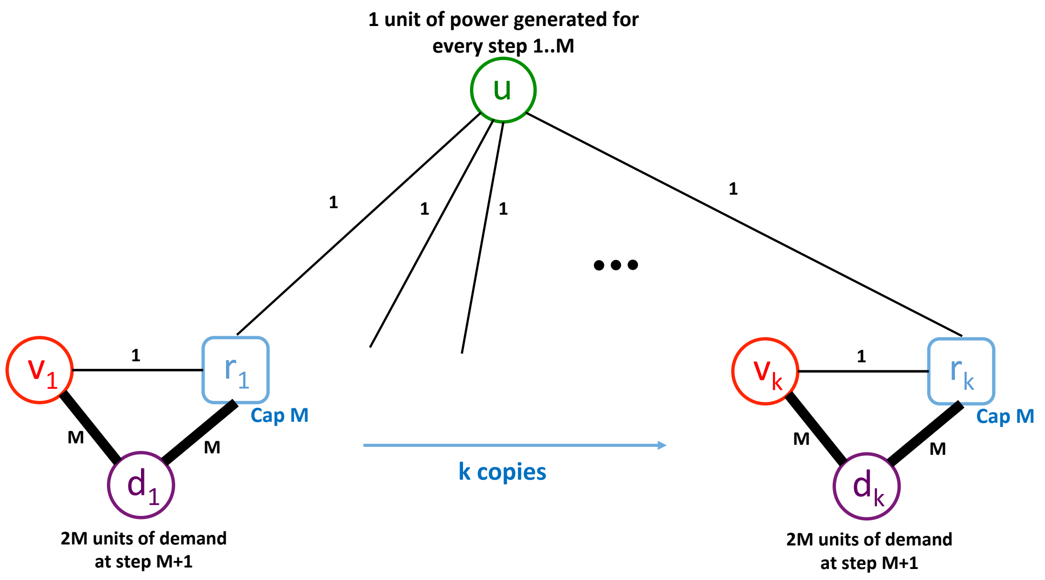

1 unit of power generated for every step 1..M
u
1
1
1
1
v1
1
r1
Cap M
M
M
d1
2M units of demand at step M+1
k copies
vk
1
rk
Cap M
M
M
dk
2M units of demand at step M+1

Figure 1 A hard example for general electrical networks with online supplies.

Let M be a large integer and let $k = \lfloor\sqrt{M}\rfloor$. In each time step $i = 1, \ldots, M$, one unit of power is generated at supply node u. The power can be stored in any of the storage units at $r_1, \ldots, r_k$ with capacity M. At time $i = M + 1$, there are $2M$ units of demand at all the demand nodes $d_1, \ldots, d_k$. The demand nodes' utility functions are linear and uniform.

Consider an arbitrary randomized online algorithm. The algorithm distributes the first M units of power onto the storage units. Let r_j be the storage unit with the least expected amount of power stored at the beginning of time step $i = M + 1$. Note the expected amount of power stored at r_j is at most $\frac{M}{k} = O(\sqrt{M})$. At time step $i = M + 1$, we assume M units of power at supply v_j and zero unit at all other supplies. Then, the algorithm can route at most $O(\sqrt{M})$ units of power in total to the demand nodes. The algorithm routes $O(\sqrt{M})$ units of power to demand d_j. Since the voltage phase angle difference between v_j and r_j can be at most 1, due to the edge of capacity 1 between them, the algorithm routes $O(\sqrt{M})$ units of newly generated power from v_j to d_j and $O(\sqrt{M})$ units of stored power from r_j to d_j. Similarly, the algorithm further routes at most 2 units of power to other demand nodes.

On the other hand, the optimal offline algorithm routes $2M$ units of power by first storing the M units of generated power from u on r_j and then routing to d_j. The resulting competitive ratio is $\Omega(\sqrt{M})$.

We construct a network with identical edges with the same susceptance and capacity. Note that we can model an edge of capacity $2c$ and susceptance $\mathcal{B}$ using four edges of capacity c and susceptance $\mathcal{B}$ arranged in the diamond shape.[12] We recursively use the diamond construction $\log M$ times to reduce the edges of capacity M to edges of capacity 1. Per an edge of capacity M, we get $4^{\log M}$ new edges and $\Theta(4^{\log M}) = \Theta(M^2)$ new nodes. Therefore, this network has $\Theta(M^{5/2})$ nodes and the lower bound becomes $\Omega(n^{1/5})$ where n is the number of nodes. ◀

[12] In the diamond shape, the four edges are arranged as (a, b), (b, d), (a, c), and (c, d).

C Missing Materials from Section 3

We provide the missing proofs from Section 3.

C.1 Uniform Utilities

We prove Lemma 7 below. We first prove the following structural lemma:

▶ **Lemma 14.** *Consider an arbitrary storage node r for which $\tau_r^i > 0$ or $\beta_r^i > 0$ at a time step i. Then, all the power stored in r at time i will be dispatched to the demands in no more than $\log_{\gamma_r}(\max(\beta_r^i, \tau_r^i))$ time steps. Furthermore, if $\tau_r^i > 0$, then the path $p^+(i, r)$ is congested in $\hat{x}^i$.*

Proof. Let $j \geq i$ denote the first step after i for which $r \in R_\phi^j$. By the recursive construction in Part II, we know that $\max(\beta_r^i, \tau_r^i) = \gamma_r^{j-i}$ and one of variables is zero. On the other hand, since $r \in R_\phi^j$, congestion does not block all the routes from r to D^j. Since we have not routed more flow from r to D^j at that time step, we know that $\hat{y}_r^j = 0$. This indeed proves the first part of the lemma. Furthermore, if $\tau_r^i > 0$, we know that r never gets full between time steps i to j, otherwise τ_r^i would have been zero. Thus the reason that the supply $s \in S^i$, which owns r, is not supplying r with more power is that $p^+(i, r)$ is congested; which completes the proof. ◀

Proof of Lemma 7. We check the feasibility of each set of dual constraints in the program $\mathbb{D}$ separately. Consider an arbitrary path $p \in P_D^+$ at time interval i. Constraint D1 enforces a lower bound of one on $\sum_{e \in p} \alpha_e + \tau_{v(p)}^{i(p)}$. We distinguish between two cases. If $v(p) \in R_\phi^i$, the constraint is satisfied in DC-Part II by setting $\tau_{v(p)}^i = 1$. Otherwise, $v(p) \in S^i \cup (R \setminus R_\phi^i)$, for which DC-Part I satisfies the constraint.

The feasibility of D2 constraints follows from Lemma 14; which shows that if $\tau_r^i > 0$ for some i and r, then $p^+(i, r)$ is congested. Therefore DC Part III satisfies D2.

Finally, the feasibility of D3 constraints follows directly from the iterative construction in Part II. ◀

C.2 Concave Utilities

Proof of Theorem 2. Under the concave utility model, scaling down all the storage and edge capacities by a factor ρ may only change the optimal solution by at worst a factor $1/\rho$. Recall that we assume $\frac{dW_d(x)}{dx} \in [w_{\min}, w_{\max}]$ for every demand. Let $\rho = \lfloor \log \Delta_w \rfloor + 1$ where $\Delta_w = \frac{w_{\max}}{w_{\min}}$. Given an instance of the general flow problem with concave utility functions and optimal solution OPT^*, we construct ρ instances of the problem with uniform utility as follows.

For every i and $d \in D^i$, let $\ell_d(q) = \inf\{x : \frac{dW_d(x)}{dx} \leq 2^q w_{\min}\}$ for every integer $q \in \{0, \ldots, \rho\}$. We note that ℓ_d is monotone non-increasing. For every $q \in \{1, \ldots, \rho\}$, we construct an uniform-utility instance in which a new node d' is connected to d with an edge (d, d') with capacity $\ell_d(q) - \ell_d(q-1)$. The node d is not a demand node anymore and instead d' is a demand node. Furthermore, we scale down all the storage and edge capacities by a factor of ρ. Let OPT_q denote the optimal solution to the q^{th} instance.

▶ **Claim 15.** $\frac{\mathsf{OPT}^*}{2\rho} \leq \sum_{q=1}^{\rho} 2^{q-1} w_{\min} \mathsf{OPT}_q \leq \mathsf{OPT}^*$.

Proof. We partition the optimal solution into ρ separate flows of power. Suppose that at every step of the algorithm, the flows of power are routed continuously. For $q \in [\rho]$, let F_q

denote the flows corresponding to units of power that are ending up satisfying a demand d during the interval that the total power received by d is between $[\ell_d(q), \ell_d(q-1))$. We note that

$$\frac{\mathsf{OPT}^*}{2} \leq \sum_q 2^{q-1} w_{\min} |F_q| \leq \mathsf{OPT}^* \ .$$

Let F'_q denote the flow obtained from F_q by reducing the flows on every edge and storage unit by a factor ρ. We note that F'_q is indeed a feasible solution for the q^{th} instance of the problem constructed above: we simply need to re-route the flow coming to d to d'. This completes the proof since $\mathsf{OPT}_q \geq |F'_q| \geq \frac{|F'_q|}{\rho}$ ◀

Now suppose we have an online algorithm with competitive ratio c for the uniform-utility model. Given a general instance, for every $q \in \{1, \dots, \rho\}$, we separately execute the uniform weight algorithm on the q-th instance of the problem constructed as above. Let ALG_q denote the solution corresponding to the q-th instance. Since all capacities are scaled down in each instance, we can route the flows in all instances simultaneously. At every time step we simply output the union of flows output by the instances of the uniform weight algorithm. The final utility is therefore at least

$$\sum_{q=1}^{\rho} 2^{q-1} w_{\min} \mathsf{ALG}_q \geq \sum_{q=1}^{\rho} 2^{q-1} w_{\min} \frac{\mathsf{OPT}_q}{c} \geq \frac{\mathsf{OPT}^*}{2c\rho} \ ,$$

which completes the proof since Theorem 1 gives an algorithm with constant c. ◀

D Missing Materials from Section 4

We provide the missing materials from Section 4.

D.1 Hardness of Online Demands

We show that online demands are hard to satisfy in the public storage setting by proving the following theorem:

Proof of Theorem 3. Our lower bound instance is a network similar to that considered in Theorem 6. Consider a network with n nodes where there exist one supply node with renewable power generators and $n-1$ demand nodes that are connected to the "root" supply node. Each demand node has a public storage unit with unit capacity and the decay factor of 1; these are the only storage units on the network. We assume the case of uniform utilities where each demand node requires at most 1 unit of power, that is, the utility functions are of the form $W(y) = y$ up to the limit $y = 1$.

At time $i = 1$, the supply node generates a unit of power. At time $i = 2$, exactly one demand node requests power while other nodes do not. Consider an arbitrary randomized online algorithm and let d be the demand node with the least expected amount of power stored at the beginning of time step $i = 2$. Note the expected amount of power stored at d is at most $O(1/n)$. We let d be the only demand node to request power at time $i = 2$, and the online algorithm routes $O(1/n)$ units of power.

However, the optimal offline algorithm knows where the demand is going to be and can always satisfy the power demand and obtain total utility of 1. Hence, a lower bound of $\Omega(n)$ on the competitive ratio follows. ◀

D.2 Concave Utilities

Proof of Claim 9. For each edge e, we show that $\alpha_e \geq \omega_e$ at all times where

$$\omega(e) := \frac{w_{\min}\gamma_{\min}}{l_{\max}}\left(e^{\frac{c}{C_e}\sum_{p\ni e}\gamma(p,e)x_p} - 1\right).$$

Initially, $\alpha_e = \omega_e = 0$. Note both α_e and ω_e increase only if some path p containing e is used for routing. Assume $\alpha_e \geq \omega_e$ and we show the inequality still holds after the updates due to routing through p, say, from supply s to demand d. Note that

$$\begin{aligned}
\frac{d\omega_e}{dx_p} &= \frac{c\gamma(p,e)}{C_e}\frac{w_{\min}\gamma_{\min}}{l_{\max}}e^{\frac{c}{C_e}\sum_{p\ni e}\gamma(p,e)x_p} \\
&= \frac{c\gamma(p,e)}{C_e}\omega_e + \frac{cw_{\min}\gamma_{\min}\gamma(p,e)}{l_{\max}C_e} \\
&\leq c\left(\frac{\gamma(p,e)\alpha_e}{C_e} + \frac{\gamma(p)W_d{}'(\lambda_d)}{l_{\max}C_e}\right) \\
&= \frac{d\alpha_e}{dx_p}.
\end{aligned}$$

Hence, α_e increases at a faster rate than ω_e and it follows that $\alpha_e \geq \omega_e$ throughout Algorithm 1.

If $\alpha_e = w_{\max}$, the dual constraints for any path p containing edge e are satisfied and α_e is not further increased. Then, $w_{\max} \geq \alpha_e \geq \omega_e$. It follows that

$$\begin{aligned}
w_{\max} &\geq \frac{w_{\min}\gamma_{\min}}{l_{\max}}\left(e^{\frac{c}{C_e}\sum_{p\ni e}\gamma(p,e)x_p} - 1\right) \\
\log\left(\frac{l_{\max}\Delta_w}{\gamma_{\min}} + 1\right) &\geq \frac{c}{C_e}\sum_{p\ni e}\gamma(p,e)x_p \\
O\left(\log\left(\frac{l_{\max}\Delta_w}{\gamma_{\min}}\right)\right)\frac{C_e}{c} &\geq \sum_{p\ni e}\gamma(p,e)x_p.
\end{aligned}$$

Since $l_{\max} = nT$ and $\gamma_{\min} = \gamma^T$, we see that

$$\sum_{p\ni e}\gamma(p,e)x_p \leq \frac{O(\log nT + \log\gamma^{-T} + \log\Delta_w)}{c}\cdot C_e \leq \frac{O(\log n + \log\gamma^{-T} + \log\Delta_w)}{c}\cdot C_e,$$

where the last inequality follows from the fact that $\log\gamma^{-T}$ dominates $\log T$. ◀

Proof of Claim 10. Let $\mathcal{D}_0 = \sum_e C_e\alpha_e$; so, $\mathcal{D} = \mathcal{D}_0 + \sum_d \widehat{W}_d(\lambda_d)$.

We first show that $\mathcal{D}_0 \leq 2c\mathcal{P}$. Initially, $\mathcal{P} = \mathcal{D}_0 = 0$. Assume we route an infinitesimal amount through path p from supply s to demand d and correspondingly update dual variables. We compute corresponding changes in $\mathcal{P}$ and $\mathcal{D}_0$. Note $d\mathcal{P} = W_d{}'(y_d)dy_d = \gamma(p)W_d{}'(y_d)dx_p$ since $dy_d = \gamma(p)dx_p$. Furthermore, $d\mathcal{D}_0 = \sum_{e\in p} C_e d\alpha_e = \sum_{e\in p} c\left(\gamma(p,e)\alpha_e + \frac{w_{\min}\gamma_{\min}}{l_{\max}}\right)dx_p \leq 2c\gamma(p)W_d{}'(\lambda_d)dx_p$. By construction, we increase y_d and λ_d at the same rate and hence, $y_d = \lambda_d$ for all demand $d \in D^*$. It follows that $d\mathcal{D}_0 \leq 2cd\mathcal{P}$ and $\mathcal{D}_0 \leq 2c\mathcal{P}$ at termination.

Since $\widehat{W}_d(z) \leq W_d(z), \forall z \geq 0$, it follows that $\sum_d \widehat{W}_d(\lambda_d) \leq \sum_d W_d(y_d) = \mathcal{P}$. Then, $\mathcal{D} = \sum_d \widehat{W}_d(\lambda_d) + \mathcal{D}_0 \leq (2c+1)\mathcal{P}$. ◀

Path Construction

We show how to check the condition in Line 4 and choose a routing path p in Line 6 in Algorithm 1 in polynomial time. More specifically, we show how to check the condition for paths in $P(s,d)$ for supply s and demand d and find a simple path p, if it exists, such that $\sum_{e\in p}\gamma(p,e)\alpha_e - \gamma(p){W_d}'(\lambda_d) < 0$. This is equivalent to finding a discounted variant of shortest path where α_e are the edge lengths and γ_e are the discount factors, from which $\gamma(p,e)$ can be defined (cf. [1]).

Given s and d, we run a backward variant of Dijkstra's algorithm starting with d with initialization $\sigma(d) = -{W_d}'(\lambda_d)$ and computing iteratively the discounted "distance" $\sigma(u) = \min_{p\in P(u,d)}\sum_{e\in p}\gamma(p,e)\alpha_e - \gamma(p){W_d}'(\lambda_d)$ and corresponding successor $\pi(u)$ for each node u. We compute σ and π in stages such that if s is in time t_1 and d is in time t_2, we process the time-copies $G^{t_2}, G^{t_2-1}, \dots, G^{t_1}$ in that order. For $t = t_2, \dots, t_1$, we iteratively initialize σ and π based on nodes in G^{t+1} via storage edges and then compute them for all nodes in G^t.

The correctness follows from the shortest discounted paths' optimality property in the time-expanded graph. Note that the shortest discounted paths have an optimal substructure property similar to that of shortest paths in that any suffix of a shortest discounted path is a shortest discounted path. If p is a shortest discounted path from u to d and $p = (u,v)\cup p'$, p' is a shortest discounted path from v to d. Otherwise, we can find a shorter discounted path from u to d via a shorter path $p'' \in P(v,d)$, since $\sum_{e\in p}\gamma(p,e)\alpha_e - \gamma(p){W_d}'(\lambda_d) = \alpha_{(u,v)} + \gamma_{(u,v)}\left(\sum_{e\in p'}\gamma(p',e)\alpha_e - \gamma(p'){W_d}'(\lambda_d)\right)$. Furthermore, a shortest discounted path's length increases monotonically within each time-copy G^t when it is extended to another shortest discounted path; in other words, there are no "negative-weight" edges in G^t. As storage edges are the only edges with discount factors and do not form cycles in the time-expanded graph, the path lengths within each time-copy are correctly computed with the backward variant of Dijkstra's algorithm which runs in polynomial time.

D.3 Concave Utilities and Convex Costs

We present the missing algorithm:

Algorithm 2 Online Algorithm for Concave Utilities and Convex Costs

```
1: Let $y_d = \sum_{p\in P(\cdot,d)}\gamma(p)x_p, \forall d$; $z_s = \sum_{p\in P(s,\cdot)}x_p, \forall s$    ▷ $y, z$ determined in terms of $x$
2: Let $\mu_s = {Q_s}'(\tau_s z_s)$ for $\tau_s = \rho_s^{1/(\rho_s-1)}, \forall s$    ▷ $\mu$ determined in terms of $z$
3: for $i = 1,\dots,T$ do
4:     for $s \in S^i$ in arbitrary order do
5:         while $P' = \{p \in P(s,\cdot) : \sum_{e\in p}\gamma(p,e)\alpha_e + \mu_s < \gamma(p){W_{d(p)}}'(\lambda_{d(p)})\} \neq \emptyset$ do
6:             Update continuously:
7:                 $p = \arg\max_{p\in P'}\gamma(p){W_{d(p)}}'(\lambda_{d(p)})$
8:                 $dx_p = 1$    ▷ Increase $x_p$ at a uniform rate
9:                 $\frac{d\lambda_{d(p)}}{dx_p} = \gamma(p)$
10:                $\frac{d\alpha_e}{dx_p} = c\left(\frac{\gamma(p,e)\alpha_e}{C_e} + \frac{\gamma(p){W_{d(p)}}'(\lambda_{d(p)})}{l_{\max}C_e}\right), \forall e \in p$    ▷ $c \geq 1$ is some parameter
11:        end while
12:    end for
13: end for
```

Proof of Claim 12. For each edge e, we show that $\alpha_e \geq \omega_e$ at all times where

$$\omega(e) := \frac{w_{\min}\gamma_{\min}}{l_{\max}} \left(e^{\frac{c}{C_e}\sum_{p\ni e}\gamma(p,e)x_p} - 1 \right).$$

Initially, $\alpha_e = \omega_e = 0$. Note both α_e and ω_e increase only if some path p containing e is used for routing. Assume $\alpha_e \geq \omega_e$ and we show the inequality still holds after the updates due to routing through path p, say, from supply s to demand d. Note that

$$\begin{aligned}\frac{d\omega_e}{dx_p} &= \frac{c\gamma(p,e)}{C_e}\frac{w_{\min}\gamma_{\min}}{l_{\max}} e^{\frac{c}{C_e}\sum_{p\ni e}\gamma(p,e)x_p} \\ &= \frac{c\gamma(p,e)}{C_e}\omega_e + \frac{cw_{\min}\gamma_{\min}\gamma(p,e)}{l_{\max}C_e} \\ &\leq c\left(\frac{\gamma(p,e)\alpha_e}{C_e} + \frac{W_d{}'(\lambda_d)\gamma(p)}{l_{\max}C_e}\right) \\ &= \frac{d\alpha_e}{dx_p}.\end{aligned}$$

Hence, α_e increases at a faster rate than ω_e and it follows that $\alpha_e \geq \omega_e$ throughout Algorithm 2.

If $\alpha_e = w_{\max}$, the dual constraints for any path p containing edge e are satisfied and α_e is not further increased. Then, $w_{\max} \geq \alpha_e \geq \omega_e$. On the same line of reasoning as in Lemma 8, we see that

$$\frac{O(\log n + \log\gamma^{-T} + \log\Delta_w)}{c} \cdot C_e \geq \sum_{p\ni e}\gamma(p,e)x_p. \qquad ◀$$

Proof of Claim 13. Let $\mathcal{D}_0 = \sum_e C_e\alpha_e + \sum_s Q_s^*(\mu_s)$; so, $\mathcal{D} = \mathcal{D}_0 + \sum_d \widehat{W}_d(\lambda_d)$. Assume $\mathcal{D}_0 \leq (2c+1)\rho^{\rho/(\rho-1)}\mathcal{P}$. Since $\widehat{W}_d(z) \leq W_d(z), \forall z \geq 0$, it would follow that $\sum_d \widehat{W}_d(\lambda_d) \leq \sum_d W_d(y_d) \leq \mathcal{P}$. Then, $\mathcal{D} = \sum_d \widehat{W}_d(\lambda_d) + \mathcal{D}_0 \leq \left((2c+1)\rho^{\rho/(\rho-1)} + 1\right)\mathcal{P}$, and the claim would follow.

We now show $\mathcal{D}_0 \leq (2c+1)\rho^{\rho/(\rho-1)}\mathcal{P}$. For $\sigma = (2c+1)\rho^{\rho/(\rho-1)}$ and $\rho > 1.44$, we show $d\mathcal{P} \geq \frac{1}{\sigma}d\mathcal{D}_0$. Initially, $\mathcal{P} = \mathcal{D}_0 = 0$. Assume we route an infinitesimal amount through path p, say, from supply s to demand d and correspondingly update primal and dual variables. We compute the resulting changes in the primal objective $\mathcal{P}$ and (partial) dual objective $\mathcal{D}_0$: $d\mathcal{P} = W_d{}'(y_d)dy_d - Q_s{}'(z_s)dz_s$; $d\mathcal{D}_0 = \sum_{e\in p} C_e d\alpha_e + (Q_s^*)'(\mu_s)d\mu_s$. Note $dz_s = dx_p$ and $dy_d = \gamma(p)dx_p$.

Note $d\mathcal{P} \geq \frac{1}{\sigma}d\mathcal{D}_0$ is equivalent to

$$W_d{}'(y_d)dy_d - Q_s{}'(z_s)dz_s \geq \frac{1}{\sigma}\left(\sum_{e\in p} C_e d\alpha_e + (Q_s^*)'(\mu_s)d\mu_s\right). \tag{6}$$

By the dual variables' updates, (6) is equivalent to

$$W_d{}'(y_d)dy_d - Q_s{}'(z_s)dz_s \geq \frac{1}{\sigma}\left(c\cdot dx_p\cdot\sum_{e\in p}\left(\gamma(p,e)\alpha_e + \frac{W_d{}'(\lambda_d)\gamma(p)}{l_{\max}}\right) + (Q_s^*)'(\mu_s)d\mu_s\right).$$

Since $\sum_{e\in p}\gamma(p,e)\alpha_e + \mu_s < \gamma(p)W_d{}'(\lambda_d)$, the right hand side is upper bounded by $\frac{1}{\sigma}\left(2c\gamma(p)W_d{}'(\lambda_d)dx_p + (Q_s^*)'(\mu_s)d\mu_s\right)$. It is sufficient to show

$$\left(1 - \frac{2c}{\sigma}\right)\gamma(p)W_d{}'(\lambda_d)dx_p - Q_s{}'(z_s)dz_s \geq \frac{1}{\sigma}(Q_s^*)'(\mu_s)d\mu_s \ .$$

Since $\sum_{e\in p}\gamma(p,e)\alpha_e+\mu_s<\gamma(p){W_d}'(\lambda_d)$, it suffices to show

$$\left(1-\frac{2c}{\sigma}\right)\mu_s-{Q_s}'(z_s)\geq\frac{1}{\sigma}(Q_s^*)'(\mu_s)\frac{d\mu_s}{dz_s}\quad. \tag{7}$$

If $Q_s(z)=a_sz^{\rho_s}$, then $Q_s^*(\mu)=\frac{\rho_s-1}{\rho_s}\frac{1}{(a_s\rho_s)^{1/(\rho_s-1)}}\mu^{\rho_s/(\rho_s-1)}$. Also, $\mu_s={Q_s}'(\tau_sz_s)$. Then, (7) reduces to $\sigma\geq\frac{{\tau_s}^{\rho_s}(\rho_s-1)}{{\tau_s}^{\rho_s-1}-1}+\frac{2c{\tau_s}^{\rho_s-1}}{{\tau_s}^{\rho_s-1}-1}$. For $\tau_s={\rho_s}^{1/(\rho_s-1)}$, the right hand side is equal to ${\rho_s}^{\rho_s/(\rho_s-1)}+\frac{2c\rho_s}{\rho_s-1}$. For $\rho_s\geq1.44$, it is upper bounded by $(2c+1){\rho_s}^{\rho_s/(\rho_s-1)}$ which is exactly the value of σ chosen. Therefore, (7) holds and $d\mathcal{P}\geq\frac{1}{\sigma}d\mathcal{D}_0$. For a ρ_0 constant smaller than 1.44, we would need to have a multiplicative factor greater than $(2c+1)$ in the penultimate step. ◀

LP-Relaxations for Tree Augmentation*

Guy Kortsarz[1] and Zeev Nutov[2]

1 **Rutgers University, Camden, NJ, USA**
guyk@camden.rutgers.edu

2 **The Open University of Israel, Ra'anana, Israel**
nutov@openu.ac.il

Abstract

In the Tree Augmentation Problem (TAP) the goal is to augment a tree T by a minimum size edge set F from a given edge set E such that $T \cup F$ is 2-edge-connected. The best approximation ratio known for TAP is 1.5. In the more general Weighted TAP problem, F should be of minimum weight. Weighted TAP admits several 2-approximation algorithms w.r.t. to the standard cut-LP relaxation. The problem is equivalent to the problem of covering a laminar set family. Laminar set families play an important role in the design of approximation algorithms for connectivity network design problems. In fact, Weighted TAP is the simplest connectivity network design problem for which a ratio better than 2 is not known. Improving this "natural" ratio is a major open problem, which may have implications on many other network design problems. It seems that achieving this goal requires finding an LP-relaxation with integrality gap better than 2, which is an old open problem even for TAP. In this paper we introduce two different LP-relaxations, and for each of them give a simple algorithm that computes a feasible solution for TAP of size at most 7/4 times the optimal LP value. This gives some hope to break the ratio 2 for the weighted case.

1998 ACM Subject Classification G.2.2 Graph Theory: Graph algorithms

Keywords and phrases Tree Augmentation, LP-relaxation, Laminar family, Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.13

1 Introduction

1.1 Problem definition and related problems

A graph (possibly with parallel edges) is *k-edge-connected* if there are k pairwise edge-disjoint paths between every pair of its nodes. We study the following fundamental connectivity augmentation problem: given a connected undirected graph $G = (V, E_G)$ and a set of additional edges (called "links") E on V disjoint to E_G, find a minimum size edge set $F \subseteq E$ such that $G + F = (V, E_G \cup F)$ is 2-edge-connected. Contracting the 2-edge-connected components of the input graph G results in a tree. Hence, our problem is:

Tree Augmentation Problem (TAP)
Instance: A tree $T = (V, E_T)$ and a set of links E on V disjoint to E_T.
Objective: Find a minimum size subset $F \subseteq E$ of links such that $T \cup F$ is 2-edge-connected.

* This work was partially supported by NSF grant number 434923.

TAP can be formulated as the problem of covering the edges of a tree by paths. For $u, v \in V$ let $(u, v) \in E_T$ denote the edge in T and uv the link in E between u and v. Let $P(uv) = P_T(uv)$ denote the path between u and v in T. A link uv *covers* all the edges along the path $P(uv)$. Then TAP is the problem of finding a minimum subset of (paths of the) links that cover the edges of T.

TAP can be also formulated as the problem of covering a laminar set family. In what follows, root T at some node r. The choice of the root r defines a partial order on V: u is a *descendant* of v (or v is an *ancestor* of u) if v belongs to $P(ru)$. The *rooted subtree* of T induced by v and its descendants is denoted by T_v (v is the root of T_v). Let $\mathcal{T} = \{T_v : v \in V \setminus \{r\}\}$. The family of node sets of the trees in $\mathcal{T}$ is laminar, and $F \subseteq E$ is a feasible solution for TAP if and only if F covers $\mathcal{T}$, namely, for every $T' \in \mathcal{T}$ there is a link in F from T' to $T \setminus T'$.

TAP is also equivalent to the problem of augmenting the edge-connectivity from k to $k+1$ for any odd k; this is since the family of minimum cuts of a k-connected graph with k odd is laminar.

In the more general Weighted TAP problem, the links in E have weights $\{w_e : e \in E\}$ and the goal is to find a minimum weight augmenting edge set $F \subseteq E$ such that $T \cup F$ is 2-edge connected. Even a more general problem is the 2-Edge-Connected Subgraph problem, where the goal is to find a spanning 2-edge-connected subgraph of a given weighted graph; Weighted TAP is a particular case, when the input graph contains a connected spanning subgraph of cost zero.

In this paper we introduce new LP-relaxations for TAP (that are also valid for Weighted TAP) and prove that their integrality gap for TAP less than 2.

1.2 Previous and related work

TAP is NP-hard even for trees of diameter 4 [9], or when the set E of links forms a cycle on the leaves of T [3]. The first 2-approximation for Weighted TAP was given 24 years ago in 1981 by Fredrickson and Jájá [9], and was simplified later by Khuller and Thurimella [15]. These algorithms reduce the problem to the Min-Cost Arborescence problem, that is solvable in polynomial time [5], while invoking a factor of 2 in the ratio. The primal-dual algorithm of [12, 11] is another combinatorial 2-approximation algorithm for the problem. The iterative rounding algorithm of Jain [13] is an LP-based 2-approximation algorithms. These algorithms achieve ratio 2 w.r.t. to the standard **cut-LP** that seeks to minimize $\sum_{e \in E} w_e x_e$ over the following polyhedron:

$$x_e \geq 0 \quad \forall e \in E \tag{1}$$

$$x(\delta(T')) \geq 1 \quad \forall T' \in \mathcal{T} \tag{2}$$

Here $\delta(T')$ is the set of links with exactly one endnode in T', $x(F) = \sum_{e \in F} x_e$ is the sum of the variables indexed by the links in F, and $\mathcal{T}$ is the set of proper rootes subtrees of T w.r.t. the chosen root r.

Laminar set families play an important role in the design and analysis of exact and approximation algorithms for network design problems, both in the primal-dual method and the iterative rounding method, c.f. [17, 12]. Weighted TAP is the simplest network design problem for which a ratio better than 2 is not known. Breaking the "natural" ratio of 2 for Weighted TAP is a major open problem in network design, that may have implications on other problems.

As a starting point, Khuller [14] in his survey on high connectivity network design problems posed as a major open question achieving ratio better than 2 for TAP. Nagamochi [19] used a novel lower bound to achieve ratio $1.875 + \epsilon$ for TAP. The sequence of papers

[7, 8, 16] introduced additional new techniques to achieve ratio 1.8 by a much simpler algorithm and analysis, and also achieved the currently best known ratio 1.5.

Several algorithms for Weighted TAP with ratio better than 2 are known for special cases. In [6] is given an algorithm with ratio $(1 + \ln 2)$ and running time $n^{f(D)}$ where D is the diameter of T. In [3] it is shown how to round a half-integral solution to the cut-LP within ratio 4/3. However, as is pointed in [3], the cut-LP LP has extreme points which are not half integral.

Studying various LP-relaxations for TAP is motivated by the hope that these may lead to breaking the ratio of 2 for Weighted TAP. Thus several paper analyzed integrality gaps of LP/SDP relaxations for the problem. Cheriyan, Karloff, Khandekar, and Koenemann [4] gave an example of a TAP instance with integrality gap 1.5 w.r.t. a standard cut-LP. For the special case of TAP when every link connects two leaves, [18] obtained ratios 5/3 w.r.t. the cut LP, ratio 3/2 w.r.t. to a strengthened “leaf edge-cover” LP, and ratio 17/12 not related to any LP. However, the analysis of [18] does not extend directly to the general TAP. Cheriyan and Gao [1] showed that the 1.8-approximation algorithm of [8] achieves its ratio 1.8 w.r.t. an SDP relaxation obtained by Lasserre tightening of a standard LP supplemented by so called “non-overlapping” constraints. Recently in [2] they improved their analysis, showing that the 1.5-approximation ratio of [16] is achievable w.r.t. this SDP. However, the SDP and the analysis in [2] are quite involved, and hence might be very hard to extend to Weighted TAP.

Finally, we mention some work on the closely related 2-Edge-Connected Subgraph problem. This problem was also vastly studied. For general weights, the best known ratio is 2 by Fredrickson and Jájá [9], which can also be achieved by the algorithms in [15] and [13]. For particular cases, better ratios are known. Fredrickson and Jájá [10] showed that when the edge weights satisfy the triangle inequality, the Christofides heuristic has ratio 3/2. For the special case when all the edges of the input graph have unit weights (the “min-size” version of the problem), the currently best known ratio is 4/3 due to Sebo and Vygen [21].

1.3 Our results

In this paper, with the help of some ideas from [7, 18, 8, 16], we introduce two simple new LP-relaxations, and prove that they have integrality gap at most 7/4 for TAP. This is the first LP-relaxation for TAP for which integrality gap less than 2 is proved. We note that our algorithms use several ideas from [7, 8], but they are *not* identical to any previous algorithm.

The dual-fitting algorithm is our main result. This is essentially a primal-dual algorithm when we allow to violate the dual constraints by a factor of 7/4. Unlike *all* previous algorithms, we do not need to compute a maximum matching on the leaves, but pick some inclusionwise maximal matching. The algorithm uses some simple local steps and is faster than all previous algorithms – the running time is $\tilde{O}(mn)$. Since the algorithm and the LP are quite simple, it has a chance to be extended to Weighted Tap. We note that one type of constrainst we use, see constraints (3), can be naturally derived from the so called “subpartiton constrains” or “Gomory cuts”, which say that k disjoint sets need at least $\lceil k/2 \rceil$ edges to cover them. The constraints (3) are derived from the case $k = 3$ when applied to the two lowest levels of the tree – leaves and stems.

2 New valid constraints

In this section we introduce new LP-relaxations for TAP and in subsequent sections prove that (for the unweighted case) they both have integrality gap 7/4. Our LP-relaxations combine some ideas from [18, 8, 16], but also use new crucial valid constraints. We need some definition to introduce these constraints.

▶ **Definition 1** (shadow, shadow-minimal cover)**.** Let $P(uv)$ denote the path between u and v in T. A link $u'v'$ is a **shadow** of a link uv if $P(u'v') \subseteq P(uv)$. A cover F of T is **shadow-minimal** if for every link $uv \in F$ replacing uv by any proper shadow of uv results in a set of links that does not cover T.

We refer to the addition of all shadows of existing links as **shadow-completion**. Shadow completion does not affect the optimal solution size, since every shadow can be replaced by some link covering all edges covered by the shadow. Thus we may assume the following:

The Shadow-Completion Assumption. The set of links E is closed under shadows.

For $A, B \subseteq V$ and $F \subseteq E$ let $\delta_F(A, B)$ denote the set of links in F with one end in A and the other end in B, and let $\delta_F(A) = \delta_F(A, V \setminus A)$ denote the set of links in F with exactly one endnode in A. The default subscript in the above notation is E. To **contract** a subtree T' of T is to combine the nodes in T' into a new node v. The edges and links with both endpoints in T' are deleted. The edges and links with one endpoint in T' now have v as their new endpoint.

▶ **Definition 2** (leaf, twin link, stem)**.** The **leaves** of T are the nodes in $V \setminus \{r\}$ that have no descendants. We denote the leaf set of T by $L(T)$, or simply by L, when the context is clear. A link $ab \in \delta(L, L)$ is a **twin link** and the least common ancestor s of a, b is a **stem** if the contraction of T_s results in a new leaf; such a, b are called **twins**. Let W denote the set of twin links, and for $e \in W$ let s_e denote the stem of e.

For $A \subseteq V$, we say that a rooted subtree T' of T is **A-closed** if there is no link in E from $A \cap T'$ to $T \setminus T'$, and T' is **A-open** otherwise.

▶ **Definition 3** (locked node, locking link, dangerous locking tree)**.** A node a (or a subtree T_a) is **locked** by a link $bb' \in \delta(L, L)$ and bb' is the **locking link of** a if (see Fig. 1(a)) the tree obtained from T by contracting T_a into the node a has a rooted proper subtree $T' = T_{r'}$ that is a-closed such that $L(T') = \{a, b, b'\}$; such minimal T' is called the **locking tree** of a (note that such locking tree is unique); a locking tree is a **dangerous locking tree** if it is as in Fig. 1(b) with the links depicted present in E; namely, a locking tree is dangerous if there exists an ordering b, b' of the locking link endnodes such that:

- The contraction of ab' does not create a new leaf.
- $ab' \in E$.
- T' is b-open.

Let $\mathcal{N}$ denote the set of non-dangerous locking trees.

Note that an ordering b, b' as in the above definition may not be unique; namely, it may be that also the contraction of ab does not create a new leaf, $ab \in E$, and T' is b'-open – see Fig. 1(c).

In what follows, let us use the following notation:

- For a stem s let $\sigma(s)$ denote the set of links in $\delta(s)$ that have an endnode not in T_s.
- For $T' \in \mathcal{N}$ let $\zeta(T')$ denote the set of links incident to some non-leaf node of T'.
- Let $\mathcal{O}_L = \{A \subseteq V : |A \cap L| \text{ is odd}\}$.
- For $x \in \mathbb{R}^E$ and $F \subseteq E$ let $x(F) = \sum_{e \in F} x_e$.

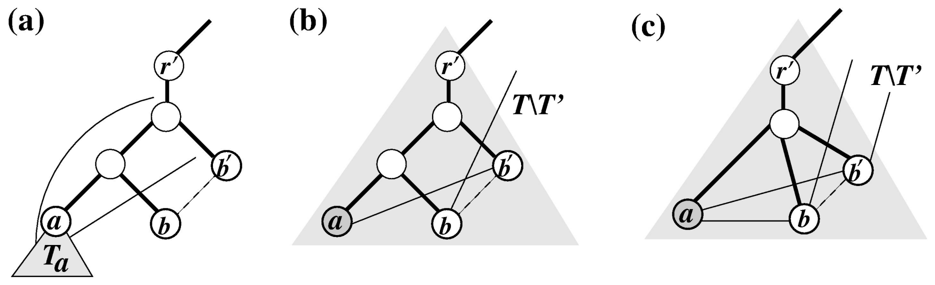


Figure 1 (a) A locking tree; no link with an endnode in T_a has its other endnode in $T \setminus T_{r'}$. (b,c) Dangerous trees; solid thin lines show links that must exist in E. The endnodes b, b' of the locking link are original leaves; in (a), a is an original leaf, and in (b),(c) the subtree T_a is contracted into a, so a may be a compound node or an original leaf. Some of the edges of T can be paths.

The proof of the following statement can be found in [8, 16]; we provide a proof sketch for completeness of exposition.

▶ **Lemma 4.** *Let F be a shadow-minimal cover of T. Then the following holds:*
(i) *$\delta_F(L, V)$ is an exact edge-cover of L, namely $|\delta_F(v)| = 1$ for every $v \in L$.*
(ii) *If $e \in F \cap W$ then $|\sigma(s_e) \cap F| = 1$.*
(iii) *$\zeta(T') \cap F \neq \emptyset$ for any $T' \in \mathcal{T}$.*

Proof. Let us say that two links **overlap** if their paths share an edge and one contains an end of the other. It is easy to see that F is not shadow minimal if and only if two links in F overlap. As any two links incident to the same leaf overlap, (i) follows.

Now let $e \in F \cap W$ and consider a link f that covers the parent edge of the stem s_e of e. It is easy to see that the only case that e and f do not overlap is if $f \in \sigma(s_e)$, and that any two links in $\sigma(s_e)$ overlap. This implies (ii).

Let T' be a locking tree as in Definition 3 (after T_a is contracted into a). We will show that if $\zeta(T) \cap F = \emptyset$ then T' is dangerous. Consider a link $e = au$ that covers the parent edge of a and a link $e' = u'v$ that covers the parent edge of T', where $v \notin T'$. Note that $e \neq e'$, since T' is a-closed. If $\zeta(T) \cap F = \emptyset$ then $\{u, u'\} \subseteq \{b, b'\}$, and since by (i) $|\delta_F(b)| = |\delta_F(b')| = 1$, we must have $\{u, u'\} = \{b, b'\}$. If $u = b$ and contraction of ab creates a new leaf, then the link in F that covers the parent edge of this new leaf belongs to $\zeta(T)$. Otherwise, T' must be dangerous, as claimed. ◀

Now we present our new valid inequalities for TAP.

▶ **Lemma 5.** *Suppose that the Shadow-Completion Assumption holds, and let x be the characteristic vector of a shadow minimal cover F of T. Then x satisfies the following constraints*

$$\begin{aligned} x(\sigma(s_e)) - x_e &\geq 0 && \forall e \in W && (3)\\ x(\zeta(T')) &\geq 1 && \forall T' \in \mathcal{T} && (4)\\ x(\delta(v)) &= 1 && \forall v \in L && (5)\\ x(\delta(A, V)) &\geq \left\lceil \frac{|A \cap L|}{2} \right\rceil && \forall A \in \mathcal{O}_L && (6) \end{aligned}$$

Proof. Consider the polyhedron Π_L defined by (1), (5), and (6). Then Π_L is the convex hull of the exact edge-covers of L, see [20, Theorem 34.2]; thus by Lemma 4(i), these constraints are valid. The validity of the constraints (3) follows from Lemma 4(ii) (in fact, $x(\sigma(s_e)) = x_e$ holds), and the validity of the constraints (4) follows from Lemma 4(iii). ◀

In subsequent sections we will consider two LPs, where both have the constraints (1), (2), and (3). One LP has an additional constraint (4), while the other LP has additional constraints (5) and (6) instead.

3 The algorithms

For a set of links $I \subseteq E$, let T/I denote the tree obtained by contracting every 2-edge-connected component of $T \cup I$ into a single node. We often refer to the contraction of every 2-edge-connected component of $T \cup I$ into a single node as the contraction of the links in I. Our algorithm iteratively contracts certain subtrees of T/I. We refer to the nodes created by contractions as **compound nodes**, and denote by C the set of compound nodes of T/I. Non-compound nodes are referred to as **original nodes** (of T). For technical reasons, the root r is also considered as a compound node, hence initially $C = \{r\}$.

Our algorithms start with a partial solution $I = \emptyset$ and with a certain matching $M \subseteq \delta(L, L) \setminus W$. We denote by U the set of leaves of T/I unmatched by M. The algorithm iteratively finds a subtree T' of T/I and a cover I' of T', and **contracts** T' **with** I', which means adding I' to I and contracting T' into a new compound node. To use the notation T/I properly, we will assume that I' is an exact cover of T', namely, that the set of edges of T/I that is covered by I' equals the set of edges of T' (this is possible due to shadow completion).

Another property of a contracted tree T' is given in the following definition.

▶ **Definition 6** (M-compatible subtree)**.** Let M be a matching on the leaves of T/I. A subtree T' of T/I is M**-compatible** if for any $bb' \in M$ either both b, b' belong to T', or none of b, b' belongs to T'. We say that a contraction of T' with I' is M-compatible if T' is M-compatible.

Assuming all compound nodes were created by M-compatible contractions, then the following type of contractions is also M-compatible.

▶ **Definition 7** (greedy contraction)**.** Adding to the partial solution I a link with both endnodes in U is called a **greedy contraction**.

One of the steps of the algorithm is to apply greedy contractions exhaustively; clearly, this can be done in polynomial time.

We now describe a more complicated type of M-compatible contractions.

▶ **Definition 8** (semi-closed tree)**.** Let M be a matching on the leaves of T/I. A rooted subtree T' of T/I is **semi-closed** (w.r.t. M) if it is M-compatible and closed w.r.t. its unmatched leaves. T' is *minimally semi-closed* if T' is semi-closed but any proper subtree of T' is not semi-closed.

For a semi-closed subtree T' of T/I let us use the following notation:

- M' is the set of links in M with both endnodes in T'.
- U' is the set of leaves of T' unmatched by M.

Our algorithms maintain the following invariant:

Algorithm 1: DUAL-FITTING($T = (V, \mathcal{E}), E$) (ratio: $\rho = 7/4$)

```
1  initialize: $I \leftarrow \emptyset$, $C \leftarrow \{r\}$.
2  $M \leftarrow$ maximal matching in $\delta(L, L) \setminus W$, $U \leftarrow$ leaves unmatched by $M$.
3  Contract every link $ab \in W$ with $a, b \in U$.
4  Exhaust greedy contractions and update $I, C$ accordingly.
5  while do
6      $T/I$ has more than one node
7  Find $T', I'$ as in Lemma 10.
8  Contract $T'$ with $I'$.
9  Exhaust greedy contractions and update $I, C$ accordingly.
10 return $I$
```

Partial Solution Invariant. The partial solution I is obtained by sequentially applying a greedy contraction or a legal semi-closed tree contraction with an exact cover.

▶ **Definition 9** (dangerous semi-closed tree)**.** A semi-closed subtree of T/I is dangerous (w.r.t. a matching M) if it is as in Definition 3 with $bb' \in M$.

In [8, 16] the following is proved:

▶ **Lemma 10** ([8, 16])**.** *Suppose that the Partial Solution Invariant hold for T, M, and I, and that T/I has no greedy contraction. Then there exists a polynomial time algorithm that finds a non-dangerous semi-closed tree T' of T/I and an exact cover $I' \subseteq E$ of T' of size $|I'| = |M'| + |U'|$.*

A formal description of the algorithms is given in Algorithms 1 and 2. Algorithm 1 and its dual-fitting analysis are our main results, since they are relatively simple and new. Our algorithms differ from previous algorithms in the matching M computed at step 2. In Algorithm 1 the matching M is only required to be *inclusion maximal*, while all previous algorithms computed a *maximum size* matching in $\delta(L, L) \setminus W$. This is a substantioal difference, since otherwise, to perform an LP-based analysis, one needs to add the constraints (6), as we will do in the analysis of Algorithm 2.

Our algorithms are supplemented by an *LP-based analysis* to achieve ratios better than 2 w.r.t. to the following two linear programs (LP1) and (LP2), where

- (LP1) is defined by the constraints (1), (2), (3), and (4).
- (LP2) is defined by the constraints (1), (2), (3), (5), and (6).

$$\textbf{(LP1)}\quad \begin{array}{lllr} \min & x(E) & & \\ \text{s.t.} & x_e \geq 0 & \forall e \in E & (1) \\ & x(\delta(T')) \geq 1 & \forall T' \in \mathcal{T} & (2) \\ & x(\sigma(s_e)) - x_e \geq 0 & \forall e \in W & (3) \\ & x(\zeta(T')) \geq 1 & \forall T' \in \mathcal{N} & (4) \end{array}$$

$$\textbf{(LP2)}\quad \begin{array}{lllr} \min & x(E) & & \\ \text{s.t.} & x_e \geq 0 & \forall e \in E & (1) \\ & x(\delta(T')) \geq 1 & \forall T' \in \mathcal{T} & (2) \\ & x(\sigma(s_e)) - x_e \geq 0 & \forall e \in W & (3) \\ & x(\delta(v)) = 1 & \forall v \in L & (5) \\ & x(\delta(A, V)) \geq \lceil |A \cap L|/2 \rceil & \forall A \in \mathcal{O}_L & (6) \end{array}$$

Algorithm 2: PRIMAL-FITTING$(T=(V,\mathcal{E}),E)$ (ratio: $\rho = 7/4$)

```
1  initialize: $C \leftarrow \{r\}$.
2  $F_L \leftarrow$ min-$w$-weight exact edge-cover of $L$, $w_e = \begin{cases} \rho & e \in \delta(L,L) \setminus W \\ \rho - \frac{1}{2} & e \in \delta(L, V \setminus L) \\ \rho + \frac{1}{2} & e \in W \end{cases}$
     $M \leftarrow \delta_{F_L}(L,L)$, $U \leftarrow$ the set leaves of $T$ unmatched by $M$
3  $I \leftarrow M \cap W$, $M \leftarrow M \setminus W$.
4  Exhaust greedy contractions and update $I, C$ accordingly.
5  while do
6      $T/I$ has more than one node
7  Find $T', I'$ as in Lemma 10.
8  Contract $T'$ with $I'$.
9  Exhaust greedy contractions and update $I, C$ accordingly.
10 return $I$
```

▶ **Theorem 11.** *Algorithm 1 computes a solution I of size at most $7/4$ times the optimal value of (LP1).*

▶ **Theorem 12.** *Algorithm 2 computes a solution I of size at most $7/4$ times the optimal value of (LP2).*

4 Dual-fitting analysis of Algorithm 1 (Theorem 11)

For a link $e \in E$ let us use the following notation:

- $\delta_{\mathcal{T}}^{-1}(e) = \{T' \in \mathcal{T} : e \in \delta(T')\}$; recall that $\mathcal{T}$ is the family of proper rooted subtrees of T.
- $\sigma_S^{-1}(e) = \{s \in S : e \in \sigma(s)\}$; recall that S is the set of stems of T.
- $\zeta_{\mathcal{N}}^{-1}(e) = \{T' \in \mathcal{N} : e \in \zeta(T')\}$; recall that $\mathcal{N}$ is the family of non-dangerous locking trees.

With this notation, the dual LP of (LP1) is:

$$\textbf{(D)} \qquad \begin{array}{lll} \max & y(\mathcal{E}) + q(\mathcal{T}) & \\ \text{s.t.} & y(\delta_{\mathcal{T}}^{-1}(e)) + z(\sigma_S^{-1}(e)) - |\{e\} \cap W| z_e + q(\zeta_{\mathcal{N}}^{-1}(e)) \le 1 & \forall e \in E \\ & y_{T'} \ge 0 & \forall T' \in \mathcal{T} \\ & z_w \ge 0 & \forall w \in W \\ & q_{T'} \ge 0 & \forall T' \in \mathcal{N} \end{array}$$

We rewrite Algorithm 1 with the updates of the dual variables as Algorithm 3.

Note that every compound node v of T/I is obtained by contracting some (not necesarly rooted) subtree T' of T, and that every compound leaf v of T/I is obtained by contracting a rooted subtree of T. Thus in the algorithm, assigning value y_v to a compound leaf v of T/I means that we assign value y_v to the subtree that was contracted into v. During the algorithm, every non-zero dual variable corresponds to some node v of T/I; if v is an original leaf then this variable is y_v, and if v is a compound node then these are the variables of subtrees contracted into v. This is so since in the "while loop" of the algorithm, immediately after some dual variable is raised, the entire subtree corresponding to this variable is contracted into a compound node.

▶ **Definition 13.** During the algorithm, the **dual load** $\mu(e)$ of a link e is defined as the sum of the dual variables in the constraint of e in the dual program, namely

$$\mu(e) = y(\delta_{\mathcal{T}}^{-1}(e)) + z(\sigma_S^{-1}(e)) - |\{e\} \cap W| z_e + q(\zeta_{\mathcal{N}}^{-1}(e)) \ .$$

Algorithm 3: Dual-Update($T = (V, \mathcal{E}), E$) (ratio: $\rho = 7/4$)

```
1  initialize: $C \leftarrow \{r\}$;
             $y \leftarrow 0$, $z \leftarrow 0$, $q \leftarrow 0$.
2  $M \leftarrow$ maximal matching in $E(L,L) \setminus W$, $U \leftarrow$ leaves unmatched by $M$.
             $y_v \leftarrow 1$ if $v \in U$, $y_v \leftarrow \rho - 1$ if $v \in L \setminus U$.
             $z_e \leftarrow \rho - 1$ for every link $e = ab \in W$ with $a, b \in U$.
3  $I \leftarrow M \cap W$, $M \leftarrow M \setminus W$.
4  Exhaust greedy contractions and update $I, C$ accordingly.
5  while do
6      $T/I$ has more than one node
7  Find $T', I'$ as in Lemma 10.
     Case 1: $|C'| = 0$ and either: $|M'| = 0$ or $|M'| = 1, |U'| \geq 2$
           $y_{T'} \leftarrow \rho - 1$
           $y_v \leftarrow \rho - 1$ if $v \in U'$ and $y_v \leftarrow 0$ if $v \in L' \setminus U'$.
     Case 2: $|C'| = 0$ and $|M'| = |U'| = 1$ (so $T' \in \mathcal{N}$)
           $q_{T'} \leftarrow \rho - 1$
8  Contract $T'$ with $I'$.
9  Exhaust greedy contractions and update $I, M, C$ accordingly.
10 return $I$
```

The **dual credit** $\pi(v)$ of a node v is defined as follows. Let $\pi'(v)$ be the sum of the dual variables y and q that correspond to v minus the number of links used by the algorithm to contract the corresponding tree into v. Then $\pi(v) = \pi'(v)$ if v does not contain r, and $\pi(v) = \pi'(v) + 1$ otherwise.

Note that the dual load of a link $e = uv$ can be written as a sum of two parts $\mu(e) = \mu_v(e) + \mu_u(e)$ where: $\mu_v(e)$ is the sum of the dual variables associated with v that contribute to $\mu(e)$, and $\mu_u(e)$ is the sum of the dual variables associated with u that contribute to $\mu(e)$.

▶ **Lemma 14.** *At the end of step 3 of the algorithm, and then at the end of every iteration in the "while" loop, the following holds.*

(i) *If a link e has exactly one endnode in a node v of T/I, then $\mu_v(e) \leq 1$, and $\mu_v(e) \leq \rho - 1$ unless the original endnode of e contained in v is an original unmatched leaf. If e has both endnodes in a compound node v then $\mu(e) \leq \rho$.*

(ii) *If $\rho \geq 1.75$ then $\pi(v) \geq 1$ for any $v \in C \cup U$.*

Proof. It is easy to see that the statement holds at the end of step 3 of the algorithm. We will prove by induction on the number of contraction steps that the statement continues to hold during the algorithm. For that, let us consider various operations performed by the algorithm.

Let us consider the greedy contraction operation. Then (i) continues to hold since greedy contractions do not change the dual variables. Suppose that a link uv was contracted into a compound node c, where u, v are leaves of T/I. By the induction hypothesis, $\pi(u), \pi(v) \geq 1$. Thus $\pi(c) \geq \pi(u) + \pi(v) - 1 \geq 1$, and hence (ii) continues to hold as well.

The other operation is contracting a semi-closed tree T' with I' into a new compound node c. The easy case is when $|C'| \geq 1$ or $|M'| \geq 2$. Then (i) continues to hold since in this case we do not change the dual variables. Also (ii) holds for c, since in this case

$$\pi(c) \geq (\pi(C') + 2(\rho - 1)|M'| + |U'|) - (|M'| + |U'|) \geq \pi(C') + |M'|/2 \geq 1 \ .$$

Now we consider the more complicated cases 1 and 2 in the "while" loop.

Case 1: $|C'| = 0$ and either $|M'| = 0$ or $|M'| = 1, |U'| \geq 2$. Recall that in this case we zero the dual variables of the endnodes of the link in M', if any, raise the dual variables of the unmatched leaves by $\rho - 1$, and raise the dual variable $y_{T'}$ of T' from 0 to $\rho - 1$. Let $e = uv$ be a link and consider two cases.

If e has exactly one endnode in T', say v, then v is not an unmatched leaf of T', since T' is semi-closed. Thus since $C' = \emptyset$, $\mu_v(e) = 0$ after changing the dual variables, and $\mu_c(e) = \rho - 1$ after we contract T' into c. Hence (i) holds for e.

Suppose that e has both endnodes in T'. Then the dual load of e can only decrease, unless one endnode of e, say v, is an unmatched leaf of T'. Note that then the other endnode of e is not an unmatched leaf, since T' has no greedy contraction. Before we change the dual variables, we have $\mu_v(e) \leq 1$, by the induction hypothesis. After we change the dual variables, $\mu_u(e) = 0$ (since T' is semi-closed, and since we zero the dual variables of the endnodes of the link in M'). Hence at the end of the operation we have $\mu(e) \leq \rho$, and e enters the new compound node c, so (i) continues to hold.

Now we show that (ii) holds for the new compound node c. Note that

$$\begin{aligned}
\pi(c) &\geq (y_{T'} + \pi(U') + (\rho-1)|U'|) - (|M'| + |U'|) \\
&\geq (\rho - 1) + |U'| + (\rho-1)|U'| - |M'| - |U'| \\
&= (\rho-1)(|U'|+1) - |M'|
\end{aligned}$$

If $|M'| = 0$ then we get $\pi(c) \geq 2(\rho - 1) = 1.5$. If $|M'| = 1$ and $|U'| \geq 2$ then we get $\pi(c) \geq 3(\rho - 1) - 1 = 1.25$. In both cases, (ii) continues to hold for c.

Case 2: $|C'| = 0$ and $|M'| = |U'| = 1$ (T' is locking non-dangerous). In this case we only raise the dual variable $q_{T'}$ to $\rho - 1$, and it is easy to verify that (i) continues to hold in this case. To see that (ii) continues to hold for the new compound node c note that

$$\begin{aligned}
\pi(c) &\geq (q_{T'} + \pi(U') + 2(\rho-1)|M'|) - (|M'| + |U'|) \\
&\geq (\rho - 1) + 1 + 2(\rho - 1) - 2 \\
&= 3(\rho - 1) - 1 = 1.25
\end{aligned}$$

This concludes the proof of the lemma. ◀

The above lemma implies that at the end of the algorithm, the dual solution (y, z, q) violates the dual constraints by a factor of ρ, and thus $(y, z, q)/\rho$ is a feasible solution to the dual program. Hence by the Weak Duality Theorem, $y(\mathcal{E}) + q(\mathcal{T}) \leq \rho\tau$, where τ is the optimal LP value. If $\rho \geq 1.75$, then the unique compound node (that contains r) has dual credit at least 1, and thus our dual solution fully pays for the links added, namely, $y(\mathcal{E}) + q(\mathcal{T}) \geq |I|$. Consequently, for $\rho = 1.75$ we get $|I| \leq y(\mathcal{E}) + q(\mathcal{T}) \leq \rho\tau$, as required.

5 Primal-fitting analysis of Algorithm 2 (Theorem 12)

5.1 Reduction to the minimum weight leaf edge-cover problem

Let Π be the polyhedron defined by the constraints of (LP1), namely:

$$\begin{array}{rclll} x_e & \geq & 0 & \forall e \in E & (1) \\ x(\delta(T')) & \geq & 1 & \forall T' \in \mathcal{T} & (2) \\ x(\sigma(s_e)) - x_e & \geq & 0 & \forall e \in W & (3) \\ x(\delta(v)) & = & 1 & \forall v \in L & (5) \\ x(\delta(A,V)) & \geq & \lceil |A \cap L|/2 \rceil & \forall A \in \mathcal{O}_L & (6) \end{array}$$

Let $\tau = \min\{x(E) : x \in \Pi\}$ be the optimal value of (LP2). Let $R = V \setminus (L \cup S)$. Let $\rho \geq 1.5$ be a paramter set later to $\rho = 7/4$. Recall the weight function w on $E(L,V)$ defined at step 2 of Algorithm 2:

$$w_e = \begin{cases} \rho & \text{if } e \in \delta(L,L) \setminus W \\ \rho - \frac{1}{2} & \text{if } e \in \delta(L, V \setminus L) \\ \rho + \frac{1}{2} & \text{if } e \in W \end{cases}$$

▶ **Lemma 15.** *Let F_L be a minimum w-weight exact edge-cover of L and $x \in \Pi$ such that $x(E) = \tau$. Then:*

$$\rho\tau \geq w(F_L) + \frac{1}{2}\sum_{v \in R} x(\delta(v)) \ . \tag{7}$$

Proof. Let Π_L be the polyhedron defined by the constraints (1), (5), and (6). Then Π_L is the convex hull of the exact edge-covers of L, see [20, Theorem 34.2]. Let x' be defined by $x'_e = x_e$ if $e \in \delta(L,V)$ and $x'_e = 0$ otherwise. Note that $x' \in \Pi_L$, since x satisfies (1), (5), and (6). Since F_L is an optimal (integral) exact cover of L with respect to the weights w_e and $x' \in \Pi_L$, we have:

$$x' \cdot w \geq w(F_L) \ .$$

Assign ρx_e tokens to every $e \in E$. The total amount of tokens is exactly $\rho x(E) = \rho\tau$. We will show that these tokens can be moved around such that the following holds:

(i) Every $e \in \delta(L,L)$, and thus every $e \in W$, keeps its initial ρx_e tokens.
(ii) Every $e \in \delta(L, V \setminus L)$ keeps $(\rho - \frac{1}{2})x_e$ tokens from its initial ρx_e tokens.
(iii) Every $v \in R$ gets $\frac{1}{2}x_e$ token for each $e \in \delta(v)$.
(iv) Every $e \in W$ gets additional $\frac{1}{2}x_e$ token, to a total of $(\rho + \frac{1}{2})x_e$ tokens.

This distribution of tokens is achieved in two steps. In the first step, for every $e \in E$, move $\frac{1}{2}x_e$ token from the ρx_e tokens of e to each non-leaf endnode of e, if any. Note that after this step, (i), (ii), and (iii) hold. In the second step, every $e \in W$ gets $\frac{1}{2}x(\sigma(s_e))$ tokens moved at the first step to its stem s_e by the links in $\sigma(s_e)$. The amount of such tokens is at least $\frac{1}{2}x_e$, by (3). This gives an assignment of tokens as claimed. ◀

To prove Theorem 12 we prove the following.

▶ **Theorem 16.** *For $\rho = 7/4$, Algorithm 2 computes a solution I of size at most the right-hand size of (7). Thus $|I| \leq \rho\tau = \frac{7}{4}\tau$.*

5.2 Analysis of the algorithm (Proof of Theorem 16)

Let $M = \delta_{F_L}(L, L)$ be the set of leaf-to-leaf links in F_L and U the set of leaves unmatched by M. Then for $\rho = 7/4$ we have:

$$w(F_L) = \rho|M \setminus W| + \left(\rho - \frac{1}{2}\right)|U| + \left(\rho + \frac{1}{2}\right)|M \cap W| = \frac{7}{4}|M \setminus W| + \frac{5}{4}|U| + \frac{9}{4}|M \cap W| \ .$$

Thus (7) implies:

$$\rho\tau \geq \frac{7}{4}|M \setminus W| + \frac{5}{4}|U| + \frac{9}{4}|M \cap W| + \frac{1}{2}\sum_{v \in R} x(\delta(v)) \ . \tag{8}$$

For the anlysis, we will assign tokens to nodes and edges of T according to the r.h.s. of (8), plus 1 extra token to (the compound node) r. Each time a contraction is performed (lines 3,4,7,8 in Algorithm 2), we assign 1 token to the compound node that results from the contraction. For example, every link $e \in M \cap W$ own 9/4 tokens, and when it is added to the partial solution I at step 3 of Algorithm 2, these 9/4 tokens pay both for the link addition and for the token assiged to the resulting compound node of T/I (and a spare of 1/4 token remains). After all links in $M \cap W$ are moved from M to I, we maintain the following invariant for the tree T/I and for links in M and nodes in R that are not yet contracted into compound nodes.

Tokens Invariant

(i) Every $e \in M \setminus W$ owns $\rho = \frac{7}{4}$ tokens.
(ii) Every non-compound leaf unmatched by M owns $\rho - \frac{1}{2} = \frac{5}{4}$ tokens.
(iii) Every compound node owns 1 token.
(iv) Every $v \in R$ owns $\frac{1}{2}x(\delta(v))$ tokens.

For a subtree T' of T/I let us use the following notation:

- M' is the set of (not yet contracted) links in M with both endnodes in T'.
- U' is the set of leaves of T' unmatched by M.
- U'_0 is the set of original (non-compound) leaves of T' unmatched by M.
- C' is the set of non-leaf compound nodes of T' (this includes r, if $r \in T'$).
- R' is the set of (not yet contracted) nodes in R that belong to T'.
- $\Sigma' = \sum_{v \in R'} x(\delta(v))$

Let $tokens(T')$ denote the amount of tokens in T'; this includes the tokens on nodes of T' and tokens of links in M with both endnodes in T', namely:

$$\begin{aligned} tokens(T') &= \frac{7}{4}|M'| + (|C'| + |U'| - |U'_0|) + \frac{5}{4}|U'_0| + \frac{1}{2}\Sigma' \\ &= \frac{7}{4}|M'| + |U'| + \frac{1}{4}|U'_0| + \frac{1}{2}\Sigma' + |C'| \end{aligned}$$

If we require not to overspend the credit provided by (8), then each time we contract T' with I' we need the following property.

▶ **Definition 17.** A contraction of T' with I' is *legal* if $tokens(T') \geq |I'| + 1$.

This means that the set I' of the links added to I and the 1 token assigned to the new compound node are paid by the total amount of tokens in T'. We do only legal contractions, which implies that at any step of the algorithm

$$|I| + tokens(T/I) \leq tokens(T) \ .$$

Thus at the last iteration, when T/I becomes a single compound node, $|I|$ is at most the right-hand side of (8).

Recall that after step 3, we have only two types of contractions of T' with I': a greedy contraction of a path by a single link between two unmatched leaves, and a contraction of a semi-closed tree with a link set of size $|I'| = |M'| + |U'|$. In the case of a greedy contraction, $\mathit{tokens}(T') \geq |U'| = 2$ while $|I'| = 1$; thus this contraction is legal. For a semi-closed subtree T' of T/I, we prove the following.

▶ **Lemma 18.** *Suppose that the Partial Solution Invariant and the Tokens Invariant hold for T, M, and I, and that T/I has no greedy contraction. Then $\mathit{tokens}(T') \geq |M'| + |U'| + 1$ holds for any non-dangerous semi-closed subtree T' of T/I.*

5.3 Proof of Lemma 18

Let T' be a semi-closed subtree of T/I w.r.t. M with root r' and node set V'. Assume that $\mathit{tokens}(T') - (|M'| + |U'|) < 1$. We will show that T' is dangerous. Note that by the Tokens Invariant:

$$\mathit{tokens}(T') - (|M'| + |U'|) = \frac{3}{4}|M'| + \frac{1}{4}|U'_0| + \frac{1}{2}\Sigma' + |C'| = \frac{1}{4}(3|M'| + |U'_0| + 2\Sigma') + |C'|$$

Since we assume that $\mathit{tokens}(T') - (|M'| + |U'|) < 1$, this immediately implies:

▶ **Lemma 19.** *$|C'| = 0$ and $3|M'| + |U'_0| + 2\Sigma' < 4$; thus $|M'| \leq 1$, and if $|M'| = 1$ then $|U'_0| = 0$ and $\Sigma' < 1/2$.*

Let us use the following additional notation:

- L' is the set of leaves of T'.
- S' is the set of (the original) stems of T'.

▶ **Lemma 20.** $|S'| = 0$.

Proof. Note that the Partial Solution Invariant implies that every stem s in T/I has exactly two leaf descendant, and they are both original leaves. Let a, b be the two leaf descendants of s, so a, b are original leaves and ab is a twin link. Since $ab \in W$, $ab \notin M'$. From the assumption that that T/I has no link greedy contraction we get that one of a, b is matched by M, as otherwise ab gives a greedy contraction. Moreover, $|M' \cap W| = 0$ and $|M'| \leq 1$ implies that $|M'| = 1$ and exactly one of a, b is matched by M. Consequently, $|M'| = |U'_0| = 1$, contradicting Lemma 19. ◀

▶ **Lemma 21.** $\Sigma' \geq |U'| + 1 - 2|M'|$.

Proof. Note that no link has both endnodes in U' (since T/I has no greedy contraction), and that $\delta(U') \cap \delta(T') = \emptyset$ (since T' is U'-closed). Thus

$$x(\delta(U') \cup \delta(T')) = \sum_{v \in U'} x(\delta(v)) + x(\delta(T')) \geq |U'| + 1 .$$

Let $e \in \delta(U')$. Then e contributes x_e to Σ', unless e is incident to a matched leaf. However, $x(\delta(b)) = 1$ for every matched leaf b, and the number of matched leaves in T' is exactly $2|M'|$. Hence $\Sigma' \geq |U'| + 1 - 2|M'|$, as claimed. ◀

▶ **Lemma 22.** *If $|M'| = 1$ then $|U'| = 1$.*

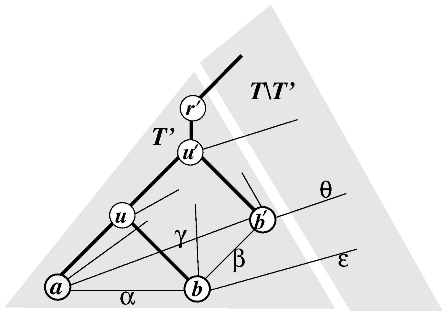


Figure 2 Illustration to the proof of Lemma 24.

Proof. If $|U'| \geq 2$ then Lemma 21 gives the contradiction $\Sigma' \geq 1$. Suppose that $|U'| = 0$. Then $|L'| = 2$, say $L' = \{b, b'\}$, and so $M' = \{bb'\}$, since $|M'| = 1$. Consequently, the contraction of bb' creates a new leaf. We obtain a contradiction by showing that then the path between b and b' in T/I has an internal compound node. By the Partial Solution Invariant b, b' are original leaves. Note that in the original tree T the contraction of bb' does not create a new leaf, since $bb' \notin W$. This implies that in T, there is a subtree $\hat{T}$ of T hanging out of a node z on the path between b and b' in T. This subtree $\hat{T}$ is not present in T/I, hence it was contracted into a compound node during the construction of our partial solution I. Thus T/I has a compound node $\hat{z}$ that contains $\hat{T}$, and since $\hat{z}$ contains a node z that belongs to the path between b and b' in T, the compound node of T/I that contains z belongs to the path between b and b'in T/I. ◀

▶ **Corollary 23.** *$|C'| = |S'| = |U'_0| = 0$, $|M'| = |U'| = 1$ (thus T' has 3 leaves), and $\Sigma' < 1/2$.*

Proof. We have $|C'| = 0$ and $|M'| \leq 1$ by Lemma 19 and $|S'| = 0$ by Lemma 20. If $|M'| = 0$ then from Lemma 21 we get that $\Sigma' \geq 2$, contradicting Lemma 19. Thus $|M'| = 1$ and by Lemmas 19 and 22 we have $\Sigma' < 1/2$, $|U'| = 1$, and $|U'_0| = 0$. ◀

We now use the properties of T' summarized in Corollary 23 to show that T' must be dangerous. Let bb' be the matched pair and a the unmatched (compound) leaf of T'. Let u and u' be the least common ancestor of ab and ab', respectively, and assume w.l.o.g. that u is a descendant of u' (see Fig. 2, and note that $u = u'$ or/and $u' = r'$ may hold). Let $x_{ab} = \alpha$, $x_{bb'} = \beta$, $x_{ab'} = \gamma$, $x(\delta(b, T \setminus T') = \epsilon$, and $x(\delta(b', T \setminus T') = \theta$.

▶ **Lemma 24.** *$\alpha, \theta > 0$ or $\gamma, \epsilon > 0$; if $u \neq u'$ then $\gamma, \epsilon > 0$.*

Proof. Consider the contribution to Σ' of links in cuts $\delta(a)$ and $\delta(T_{r'})$:
(i) Cut $\delta(a)$: $\frac{1}{2} > \Sigma' \geq x(\delta(a)) - (\alpha + \gamma) \geq 1 - (\alpha + \gamma)$; hence $\alpha + \gamma > \frac{1}{2}$.
(ii) Cut $\delta(T_{r'})$: $\frac{1}{2} > \Sigma' \geq x(\delta(T_{r'})) - (\theta + \epsilon) \geq 1 - (\theta + \epsilon)$; hence $\theta + \epsilon > \frac{1}{2}$.

In particular, we cannot have $\alpha, \gamma = 0$ or $\theta, \epsilon = 0$. We show that each one of the cases $\alpha, \epsilon = 0$ or $\gamma, \theta = 0$ is also not possible.

If $\alpha, \epsilon = 0$ then $\gamma, \theta > \frac{1}{2}$, giving the contradiction $1 = x(\delta(b')) \geq \gamma + \theta > 1$.

If $\gamma, \theta = 0$ then $\alpha, \epsilon > \frac{1}{2}$, giving the contradiction $1 = x(\delta(b)) \geq \alpha + \epsilon > 1$.

Now let us consider the case $u \neq u'$. Then by considering the cut $\delta(T_u)$ we get: $1/2 > \Sigma' \geq x(\delta(T_u)) - (\beta + \gamma + \epsilon) \geq 1 - (\beta + \gamma + \epsilon)$; hence $\beta + \gamma + \epsilon > 1/2$.

If $\gamma = 0$ then $\beta + \epsilon > 1/2$, and $\alpha > 1/2$ by (i); by considering the cut $\delta(b)$ we get the contradiction $1 = x(\delta(b)) \geq \alpha + \beta + \epsilon > 1/2 + 1/2 = 1$.

If $\epsilon = 0$ then $\beta + \gamma > 1/2$, and $\theta > 1/2$ by (ii); by considering the cut $\delta(b')$ we get the contradiction $1 = x(\delta(b')) \geq \beta + \gamma + \theta > 1/2 + 1/2 = 1$. ◀

Lemma 24 implies that T' is dangerous. Indeed, if $u \neq u'$, then $\gamma > 0$ implies that the link ab' exists, and $\epsilon > 0$ implies that T' is b-open. Thus, by the definition, T' is dangerous. The same holds if $u = u'$ and $\gamma, \epsilon > 0$. If $u = u'$ and $\alpha, \theta > 0$, then ab exists (since $\alpha > 0$) and T' is b'-open (since $\theta > 0$); thus by exchanging the roles of b, b' we get that T' is dangerous, by the definition.

This concludes the proof of Lemma 18.

References

1 J. Cheriyan and Z. Gao. Private communication. 2014.

2 J. Cheriyan and Z. Gao. Approximating (unweighted) tree augmentation via lift-and-project, part II. Manuscript, 2015.

3 J. Cheriyan, T. Jordán, and R. Ravi. On 2-coverings and 2-packing of laminar families. In *ESA*, pages 510–520, 1999.

4 J. Cheriyan, H. Karloff, R. Khandekar, and J. Koenemann. On the integrality ratio for tree augmentation. *Operation Research Letters*, 36(4):399–401, 2008.

5 Y. Chu and T. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.

6 N. Cohen and Z. Nutov. A $(1 + \ln 2)$-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius. *Theoretical Computer Science*, 489-490:67–74, 2013.

7 G. Even, J. Feldman, G. Kortsarz, and Z. Nutov. A 3/2-approximation for augmenting a connected graph into a two-connected graph. In *APPROX*, pages 90–101, 2001.

8 G. Even, J. Feldman, G. Kortsarz, and Z. Nutov. A 1.8-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Transactions on Algorithms*, 5(2), 2009.

9 G. N. Frederickson and J. Jájá. Approximation algorithms for several graph augmentation problems. *SIAM J. Computing*, 10:270–283, 1981.

10 G. N. Frederickson and J. Jájá. On the relationship between the biconnectivity augmentation and traveling salesman problem. *Theoretical Computer Science*, 19(2):189–201, 1982.

11 M. Goemans, A. Goldberg, S. Plotkin, E. Tardos D. Shmoys, and D. Williamson. Improved approximation algorithms for network design problems. In *SODA*, pages 223–232, 1994.

12 M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Computing*, 24(2):296–317, 1995.

13 K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

14 S. Khuller. Approximation algorithms for finding highly connected subgraphs (chapter 6). In *Approximation algorithms for NP-hard problems (Ed. D. S. Hochbaum)*. PWS, Boston, 1996.

15 S. Khuller and R. Thurimella. Approximation algorithms for graph augmentation. *J. of Algorithms*, 14:214–225, 1993.

16 G. Kortsarz and Z. Nutov. A simplified 1.5-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. To appear in *Transactions on Algorithms*, 2014.

17 L. C. Lau, R. Ravi, and M. Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge University Press, 2011.

18 Y. Maduel and Z. Nutov. Covering a laminar family by leaf to leaf links. *Discrete Applied Mathematics*, 158(13):1424–1432, 2010.

19 H. Nagamochi. An approximation for finding a smallest 2-edge connected subgraph containing a specified spanning tree. *Discrete Applied Math.*, 126:83–113, 2003.

20 A. Schrijver. *Combinatorial Optimization, Polyhedra and Efficiency.* Springer-Verlag Berlin, Heidelberg New York, 2004.

21 A. Sebo and J. Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, 34(5):597–629, 2014.

A Bi-Criteria Approximation Algorithm for k-Means*

Konstantin Makarychev[1], Yury Makarychev[2], Maxim Sviridenko[3], and Justin Ward[4]

1 **Microsoft Research, Redmond, WA, USA**
`komakary@microsoft.com`
2 **Toyota Technological Institute at Chicago, Chicago, IL, USA**
`yury@ttic.edu`
3 **Yahoo Labs, New York, NY, USA**
`sviri@yahoo-inc.com`
4 **Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland**
`justin.ward@epfl.ch`

Abstract

We consider the classical k-means clustering problem in the setting of bi-criteria approximation, in which an algorithm is allowed to output $\beta k > k$ clusters, and must produce a clustering with cost at most α times the to the cost of the optimal set of k clusters. We argue that this approach is natural in many settings, for which the exact number of clusters is a priori unknown, or unimportant up to a constant factor. We give new bi-criteria approximation algorithms, based on linear programming and local search, respectively, which attain a guarantee $\alpha(\beta)$ depending on the number βk of clusters that may be opened. Our guarantee $\alpha(\beta)$ is always at most $9+\epsilon$ and improves rapidly with β (for example: $\alpha(2) < 2.59$, and $\alpha(3) < 1.4$). Moreover, our algorithms have only polynomial dependence on the dimension of the input data, and so are applicable in high-dimensional settings.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases k-means clustering, bicriteria approximation algorithms, linear programming, local search

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.14

1 Introduction

The k-means clustering problem is one of the most popular models for unsupervised machine learning. The problem is formally defined as follows.

▶ **Definition 1.** In the *k-means* problem, we are given a set X of n points $x_1, \dots, x_n$ in $\mathbb{R}^p$ and an integer parameter $k \geq 1$. Our goal is to partition X into k clusters $S_1, \dots, S_k$ and assign each cluster a center a_i so as to minimize the cost $\sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - a_i\|^2$.

The most common heuristic for k-means is Lloyd's algorithm introduced in 1957 [22, 23]. Lloyd's algorithm starts with some initial solution and then iteratively improves it by alternating two steps: at the first step, the algorithm picks the optimal clustering for the current set of centers; at the second step, the algorithm picks the optimal set of centers for

* Work partially supported by EPSRC grant EP/J021814/1 and ERC Starting Grant 335288-OptApprox.

the current clustering. While we know that the algorithm performs well on well-clusterable data [26] it performs arbitrarily badly on general instances. There exist many variants of this algorithm and many heuristics for picking the initial solution. Unfortunately, none of them give a constant (not depending on k) factor approximation. One of the most popular ones is the k-means++ algorithm that has an $O(\log k)$-approximation factor [5].

There are several results showing that k-means is NP-hard even in restricted special cases [3, 24, 13]. The general k-means clustering problem has recently been shown to be APX-hard, ruling out a PTAS in the general case [7]. However, a variety of PTASes exist for special cases of the problem. Inaba, Katoh, and Imai [16] gave a $(1 + \varepsilon)$-approximation algorithm for the case in which the number of clusters, k, and the dimension of the space, p, are fixed. Since then many more PTASes were proposed for other special cases. Most recently, PTASes have been obtained via local search algorithms in the general setting in which distances come from a metric of constant doubling dimension [15] or from a graph with forbidden minors [12]. Both of these results can be specialized yield PTASes in the standard, Euclidean setting considered here whenever the dimension of the space is fixed.

In the general case, in which the dimension is not fixed, the best constant factor approximation algorithm was proposed by Kanungo et al. [18]. Their algorithm gives $9 + \varepsilon$ factor approximation. Previously, Jain and Vazirani [17] gave a (larger) constant-factor approximation for a discrete variant of k-means, via the primal-dual method. Using and connection with the k-median problem, Anagnostopoulos, Dasgupta, and Kumar [4] also designed a constant factor approximation algorithm for the general *co-clustering problem*, which includes k-means as a special case. Aggarwal, Deshpande, and Kanan [2] showed that running the k-means++ algorithm for more steps gives an $\alpha = 4 + \varepsilon$ factor approximation by opening $\lceil 16(k + \sqrt{k}) \rceil$ centers, and also showed how to modify the resulting solution to obtain a set of k centers attaining an $O(1)$ factor guarantee.

In most practical applications the target number k of clusters is not fixed in advance. Rather, we would like to find a number k that provides a well-clusterable solution. Here, we show how to substantially improve the approximation factor by slightly violating the constraint on the number of clusters. We present bi-criteria approximation algorithms for the general case of the problem. A (β, α) bi-criteria approximation algorithm finds a solution with βk clusters, whose cost is at most α times the optimal cost of a solution using k clusters. In contrast to the approach of Aggarwal, Deshpand, and Kanan [2], our algorithms find an approximate solution for every $\beta > 1$. Our approximation is always at most 9, and decreases rapidly with β. In particular, we obtain a 4-approximation by opening only $1.65k$ centers, improving over previous results [2] by a factor of nearly 10, and obtain improved approximation factors $\alpha(\beta)$ as β continues to grow. For example, $\alpha(1.3) < 6.45$, $\alpha(1.5) < 4.8$; $\alpha(2) < 2.59$, and $\alpha(3) < 1.4$. In general, we argue that in many applications the number of clusters is not important as long as it approximately equals k. For these applications we can obtain an approximation factor very close to 1.

We give three bi-criteria algorithms – two based on linear programming and one based on local search. We show the algorithms' approximation factors as a function of β in Figure 1. Note that our linear programming algorithm attains a better approximation α for large β, while the local search algorithm is better for β near 1.

Both of our algorithms are based on a reduction from the general k-means problem, in which cluster centers may be placed at any point in $\mathbb{R}^p$, to the following problem, in which we are restricted to a given, discrete set of candidate cluster centers with specified distances from each point. As part of reduction, we utilize dimensionality reduction to ensure that the number of discrete candidate centers that must be considered is polynomial in both the number of points n and in the dimension p.

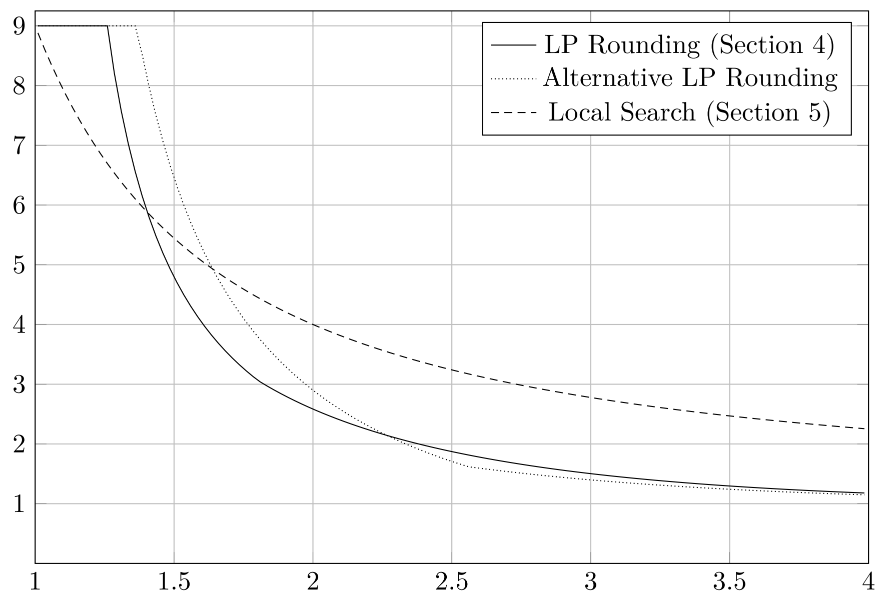


Figure 1 Approximation ratios obtained from opening βk centers.

▶ **Definition 2.** In the *k-median problem*, we are given a set of points $\mathcal{D}$, a set of potential center locations $\mathcal{C}$ and a distance function[1] $(d(i,j))_{i \in \mathcal{C}, j \in \mathcal{D}}$. The cost of assigning point j to center i is $d(i,j)$. Our goal is to open at most k centers and assign each point to a center so as to minimize the total cost.

The first approximation algorithms for the k-median problem were given by Lin and Vitter [21], who gave an LP-rounding algorithm that attains an approximation factor of $1+\varepsilon$ by opening $O(k \ln n)$ centers (i.e. a $(1+\epsilon, O(\ln n))$ bi-criteria approximation). In further work, Lin and Vitter [20] showed that if the distance function d is a *metric*, it is possible to obtain a $2(1+\varepsilon)$ approximation algorithm by opening only $(1+1/\varepsilon)k$ centers. The first constant-factor approximation for the metric k-median problem using only k centers was obtained by Arya et al. [6], who showed that a simple local search algorithm gives a $3+\varepsilon$ approximation. This remained the state of the art until recently, when Li and Svensson [19] gave a $2.732+\varepsilon$ approximation algorithm based on LP rounding. Subsequently, this has been improved to $2.675+\varepsilon$ by Byrka [8].

Unfortunately, our resulting k-median instance is non-metric, and so we must employ an alternative to the standard triangle inequality in our analysis. In the case of our LP-based algorithms, we use the fact that our reduction produces instances satisfying a 3-relaxed 3-hop triangle inequality, a concept that we define in Section 2. In the case of local search, we note that given any partition of points of $\mathbb{R}^p$ into clusters $S_1, \ldots, S_k$, the optimal location of each k-means cluster S_i's center is the centroid of all points in S_i. This, combined with the fact that our reduction to k-median approximately preserves the k-means cluster costs allows us to employ a similar approach to that of Kanungo et al. [18].

[1] Here, and throughout, we do not require the distance function to be a metric, as in some alternative definitions of the k-median problem. In particular, in our k-median instance, the distances will be the *squared* distances from given k-means instance.

1.1 Our Results

We give three approximation algorithms. The first algorithm is based on linear programming. It gives an

$$\alpha_1(\beta) = 1 + e^{-\beta}\Big(\frac{6\beta}{1-\beta} + \frac{(\beta-1)^2}{\beta}\Big)$$

approximation (see also (14) for a slightly tighter bound). The second algorithm is based on local search. It gives an

$$\alpha_2(\beta) = (1 + O(\varepsilon))\Big(1 + \frac{2}{\beta}\Big)^2$$

approximation. The third algorithm is also based on linear programming. It gives an

$$\alpha_3(\beta) = \max\big(1 + 8e^{-\beta}, \frac{\beta(e^{-1} + 8e^{-\beta})}{\beta - 1}\big)$$

approximation. The algorithm is similar to the first algorithm, but it uses pipage rounding (see [1]) instead of randomized rounding. In the conference version of the paper, we omit the description of the third algorithm. The approximation factors are shown in Figure 1.

In Section 2, we introduce the notation that we shall use throughout the rest of the paper and review standard notions related to both the k-means and k-median problems. In Section 3, we give the details of our reduction to the k-median problem. Finally, in Sections 4 and 5, respectively, we present our main LP-based algorithm and local search algorithm for the resulting k-median instances. For the sake of presentation, we defer some technical details to the appendix.

2 Preliminaries

We now fix some notation, and recall some basic properties of k-means solutions and the standard linear program for the k-median problem. Additionally, we define the notion of an α-relaxed 3-hop triangle inequality, which will be crucial to the analysis of our LP-rounding algorithms.

2.1 k-means

Consider a given instance of the k-means problem, specified by a set of points $X \in \mathbb{R}^p$. Given a partition $S = \langle S_1, \ldots, S_k\rangle$ of X and a set $C = \langle c_1, \ldots, c_k\rangle$ of centers in $\mathbb{R}^p$, denote by $\mathrm{cost}_X(S, C)$ the total cost of the clustering that, for each $1 \le i \le k$ assigns each point of S_i to the center c_i:

$$\mathrm{cost}_X(S, C) = \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|^2.$$

Note that to describe an optimal solution to the k-means problem, it is sufficient to specify either all clusters or all centers in the solution. Indeed, given a list of clusters $S_1, \ldots, S_k$, we can find the optimal assignment of centers c_i for it: the optimal choice of center c_i for S_i is $\frac{1}{|S_i|}\sum_{x_j \in S_i} x_j$. For this choice of c_i, we have

$$\sum_{x \in S_i} \|x - c_i\|^2 = \frac{1}{2|S_i|} \sum_{x', x'' \in S_i} \|x' - x''\|^2. \tag{1}$$

Given a partition $S = \langle S_1, \dots, S_k \rangle$ of X into clusters, we then denote by $\mathrm{cost}_X(S)$ the cost of this optimal choice of centers. That is,

$$\mathrm{cost}_X(S) = \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{x',x'' \in S_i} \|x' - x''\|^2.$$

Similarly, given a list C of centers $c_1, \dots, c_k$, we can find the optimal partition $S = \langle S_1, \dots, S_k \rangle$ of X into clusters. For each $c \in C$, let $N_C(c)$ be the set of those points $x \in X$ that are closer to c_i than to other centers $c_j \neq c_i$ (if a point x is at the same distance from several centers, we break the ties arbitrarily). The optimal partition for C then sets $S_i = N_C(c_i)$. Given a set C of k centers, we define $\mathrm{cost}_X(C) \equiv \mathrm{cost}_X(T)$, where $T = \langle N_C(c_1), \dots, N_C(c_k) \rangle$ is the partition induced by C.

2.2 k-median

We will reduce a given instance X of the k-means problem to an instance of the (non-metric) discrete k-median problem, specified by $\langle \mathcal{D}, \mathcal{C}, d \rangle$. By analogy with the k-means problem, we can consider a partition $S = S_1, \dots, S_k$ of points from $\mathcal{D}$, and then consider the best choice of a single center for each partition. We denote the cost of this choice by $\mathrm{cost}_{\mathcal{D},d}(S)$:

$$\mathrm{cost}_{\mathcal{D},d}(S) = \sum_{i=1}^{k} \min_{x \in \mathcal{C}} \sum_{j \in S_i} d(x, j).$$

Similarly, given a list of k centers $C = \langle c_1, \dots, c_k \rangle$, let $N_C(c_i)$ be the set of those points $x \in \mathcal{D}$ that are closer (according to the distance function d) to c_i than to any other center in C (again, if a point x is at the same distance from several facilities, we break ties arbitrarily). As in the case of k-means, we define $\mathrm{cost}_{\mathcal{D},d}(C) \equiv \mathrm{cost}_{\mathcal{D},d}(T)$ where $T = \langle N_C(c_1), \dots, N_C(c_k) \rangle$ is the partition of $\mathcal{D}$ induced by C.

Although the distance function d in our k-median instances will not satisfy the standard triangle inequality, we can show that it satisfies a relaxed variant of the following sort:

▶ **Definition 3.** We say that d satisfies an *α-relaxed 3-hop triangle inequality* on $\mathcal{D} \cup \mathcal{C}$ if, for any $j, j' \in \mathcal{D}$ and $i, i' \in \mathcal{C}$, we have

$$d(i, j) \leq \alpha \left(d(i, j') + d(i', j') + d(i', j) \right).$$

Specifically, we shall show that the distances produced by our reduction satisfy a 3-relaxed 3-hop triangle inequality.

3 Reduction from k-means to k-median

We now give the details of our reduction from the k-means to the k-median problem. In the k-median problem, a finite set $\mathcal{C}$ of candidate centers is specified, while in the k-means problem, the ideal center for each cluster S_i of points is given by the centroid of S_i. Ideally, we want to ensure that for every possible centroid of the original k-means instance, there is some nearby candidate center in $\mathcal{C}$. The following notion of an ε-approximate centroid set, introduced by[2] Matoušek [25], captures this requirement.

[2] Matoušek's original definition requires that $\mathcal{C}$ contains some point in an *ϵ-tolerance ball* centered at the centroid of each non-empty cluster of points from X. The condition presented here follows from this one (see, for example, the proof of Lemma 4.1, in [25]).

▶ **Definition 4.** A set of points $\mathcal{C} \subset \mathbb{R}^p$ is an ε-approximate centroid set for $X \subset \mathbb{R}^p$ if for every $S \subset X$,

$$\min_{c\in\mathcal{C}} \sum_{x\in S} \|x-c\|^2 \le (1+\varepsilon) \min_{c\in\mathbb{R}^p} \sum_{x\in S} \|x-c\|^2.$$

Observe that if $\mathcal{C}$ is an ε-approximate centroid set for X, then for every set of k centers C (in particular, for the optimal set C^*), there exists a k-point subset $\widetilde{C} \subset \mathcal{C}$ such that

$$\mathrm{cost}_X(\widetilde{C}) = \sum_{x\in X} \min_{c\in\widetilde{C}} \|x-c\|^2 \le (1+\varepsilon) \sum_{x\in X} \min_{c\in C} \|x-c\|^2 = (1+\varepsilon)\,\mathrm{cost}_X(C).$$

Thus, if we restrict our search for k center points in the k-means problem to only those points of $\mathcal{C}$, we lose at most a factor of $(1+\varepsilon)$.

Matoušek showed that for every set X in $\mathbb{R}^p$ and $\varepsilon > 0$, there exists an ε-approximate centroid set of size $O(|X|\varepsilon^{-p}\log(1/\varepsilon))$.

▶ **Theorem 5** (Theorem 4.4 in [25]). *Given an n-point set $X \subset \mathbb{R}^p$ and $\varepsilon > 0$, an ε-approximate centroid set for X of size $O(n\varepsilon^{-p}\log(1/\varepsilon))$ can be computed in time $O(n\log n + n\varepsilon^{-p}\log(1/\varepsilon))$.*

Unfortunately, in our setting, the dimension p of the space in which points $x_1, \dots, x_n$ lie may be as large as n. Thus, in order to apply Theorem 5, we first embed X into a low-dimensional space using the Johnson–Lindenstrauss transform.

▶ **Theorem 6** (Johnson–Lindenstrauss Flattening Lemma). *For every set of points X in $\mathbb{R}^p$ and $\varepsilon \in (0,1)$, there exists a map φ of X into $\tilde{p} = O(\log|X|/\varepsilon^2)$ dimensional space such that*

$$\|x-y\|_2^2 \le \|\varphi(x)-\varphi(y)\|_2^2 \le (1+\varepsilon)\|x-y\|_2^2. \tag{2}$$

We say that the map φ is a dimension reduction transform *for X.*

Given an instance X of k-means, we apply the dimension reduction transform to X, get a set $X' \subset \mathbb{R}^{\tilde{p}}$, and then find an ε-approximate centroid set $\mathcal{C}$ to X'. We obtain an instance $\langle X', \mathcal{C}, d\rangle$ of k-median with the squared Euclidean distance d. We show in Theorem 7 that the value of this instance is within a factor of $(1+\varepsilon)$ of the value of instance X of k-means, and, moreover, that there is a one-to-one correspondence between solutions of instance $\langle X', \mathcal{C}, d\rangle$ and solutions of instance X. We defer the proof of Theorem 7 to Appendix A.

▶ **Theorem 7.** *The following hold:*

1. *For every $\varepsilon \in (0, 1/2)$, there exists a polynomial-time reduction from k-means to k-median with distance function that satisfies the 3-relaxed 3-hop triangle inequality. Specifically, given an instance X of k-means, the reduction outputs an instance $\langle \mathcal{D}, \mathcal{C}, d\rangle$ of k-median with $|\mathcal{D}| = |X|$, $|\mathcal{C}| = n^{O(\log(1/\varepsilon)/\varepsilon^2)}$, and distance d that satisfies the 3-relaxed 3-hop triangle inequality such that*

   $$\mathrm{OPT}_X \le \mathrm{OPT}_{\langle\mathcal{D},d\rangle} \le (1+\varepsilon)\mathrm{OPT}_X,$$

   where OPT_X is the value of the optimal solution to X and $\mathrm{OPT}_{\langle\mathcal{D},d\rangle}$ is the value of the optimal solution to $\langle\mathcal{D},\mathcal{C},d\rangle$. The reduction also gives a one-to-one correspondence $\psi : \mathcal{D} \to X$ such that

   $$\mathrm{cost}_X(\psi(S)) \le \mathrm{cost}_{\mathcal{D},d}(S) \le (1+\varepsilon)\,\mathrm{cost}_X(\psi(S)),$$

   where $S = \langle S_1, \dots, S_k\rangle$ is a partition of $\mathcal{D}$ and $\psi(S) = \langle \psi(S_1), \dots, \psi(S_k)\rangle$ is the corresponding partition of X. The reduction runs in time $n^{O(\log(1/\varepsilon)/\varepsilon^2)}$.

2. *In instance $\langle \mathcal{D}, \mathcal{C}, d\rangle$, $\mathcal{C} \subset \mathbb{R}^{\tilde{p}}$ (for some $\tilde{p}$), $\mathcal{C}$ is an $(\varepsilon/3)$-approximate centroid set for $\mathcal{D}$, and $d(c,x) = \|c-x\|^2$.*

We remark, briefly, some elements of our reduction may be improved by using more sophisticated approaches for constructing approximate centroids and core sets (e.g. [14]), as well as recent specialized dimensionality reduction techniques [11]. Here we have chosen instead to present a more straightforward reduction.

4 Algorithm for k-Median with Relaxed Triangle Inequality

We now turn to the problem of approximating the k-median instance from Theorem 7. Our first algorithm is based on the following standard linear programming relaxation for the k-median problem:

$$\min \sum_{c\in\mathcal{C}} \sum_{x\in X} z_{xc} d(x,c), \tag{3}$$

$$\sum_{c\in\mathcal{C}} y_c = k, \tag{4}$$

$$\sum_{c\in\mathcal{C}} z_{xc} = 1, \qquad \forall x \in X, \tag{5}$$

$$z_{xc} \le y_c, \qquad \forall c \in \mathcal{C}, j \in X, \tag{6}$$

$$z_{xc}, y_c \ge 0. \tag{7}$$

In the integral solution, each variable y_c indicates whether the center c is open; and each variable z_{xc} indicates whether the point x is assigned to the center c. Constraint (4) asserts that we should open exactly k centers; constraint (5) ensures that every point is assigned to exactly one center; finally, constraint (6) says that points can be assigned only to open centers. In a fractional LP solution, all z_{xc} and y_c lie in the interval $[0,1]$. Note that in the integral solution, $z_{xc} = y_c$, if $z_{xc} > 0$ (as both z_{xc} and y_c must be equal to 1). We can slightly change any feasible LP solution so it also satisfies this property. Specifically, we split any center c which does not satisfy $y_c = z_{xc}$ (for some $x \in X$) into two co-located centers c_1 and c_2: one with weight z_{xc} and the other with weight $y_c - z_{xc}$. We distribute the weights $z_{x'c}$ among them as follows: we let $z_{x'c_1} = \min(z_{x'c}, y_{c_1})$; $z_{x'c_2} = y_{c_2} - \min(z_{x'c}, y_{c_1})$. Note that this is a standard assumption in the k-median literature. We refer the reader to [27] (see Lemma 1) and [10] for more details. The values y_c define the measure y on $\mathcal{C}$: $y(C) = \sum_{c\in C} y_c$. In the rounding algorithm and in the analysis, it will be convenient to think of this measure as a "continuous measure": That is, if needed we will split the centers into co-located centers to ensure that we can find a set of any given measure μ.

For every point $x \in X$, let $C_x = \{c \in \mathcal{C} : z_{xc} > 0\}$. The set C_x contains all centers that serve x in the LP solution. Recall that we modify the solution so that $y_c = z_{xc}$ if $z_{xc} > 0$. Hence, $y_c = z_{xc}$ if $x \in C_x$. For every point $x \in X$, we define its LP radius R_x as:

$$R_x = \sum_{c\in\mathcal{C}} z_{xc} d(x,c) = \sum_{c\in C_x} y_c d(x,c).$$

Observe, that the LP value, which we denote by LP, equals $\sum_{x\in X} R_x$.

Algorithm. We now describe our LP-rounding algorithm for the k-median problem with relaxed 3-hop triangle inequality.

▶ **Theorem 8.** *There exists a (β, α) bi-criteria approximation algorithm for k-means with*

$$\alpha(\beta) = 1 + e^{-\beta}\Big(\frac{6\beta}{1-\beta} + \frac{(\beta-1)^2}{\beta}\Big) \tag{8}$$

for every $\beta > 1$.

The algorithm first solves the LP problem and modifies the LP solution as described above if necessary. Then, it partitions all centers into βk groups $Z \in \mathcal{Z}$, each with LP measure $1/\beta$. It picks one center c at random from each group Z with probability βy_c (note that $\sum_{c \in Z} \beta y_c = 1$). The algorithm outputs the set of βk chosen centers, and assigns every point to the closest center.

We now describe the construction of $\mathcal{Z}$ in more detail. We partition centers into βk groups as follows. For every $x \in X$, we find the unique ball B_x around x whose LP weight exactly equals $1/\beta$ (To do so, we may split some centers, and pick some centers in B_x at the boundary of the ball but not the others). We find a subset of points $\mathcal{W}$ such that balls B_x with $x \in \mathcal{W}$ are disjoint, and for every point $x \in X$, we also define a "witness" $w(x) \in \mathcal{W}$. To this end, we sort all points $x \in X$ by the LP radius R_x in the ascending order, and then consider them one by one. For each $x \in X$, if B_x is disjoint from all previously chosen balls, then we add x to the set $\mathcal{W}$ and set $w(x) = x$. Otherwise, if B_x intersects some other ball $B_{x'}$ that is already chosen, we discard B_x and set $w(x) = x'$. If there are several balls $B_{x'}$ intersecting B_x, we pick the first x' according to our ordering as the witness. Note, that $R_{w(x)} \leq R_x$ for all x. Once we have found a disjoint collection of balls $\{B_x : x \in \mathcal{W}\}$, we add them to the set $\mathcal{Z}$. We partition centers not covered by $\cup_{x \in \mathcal{W}} B_x$ into groups of LP weight $1/\beta$ arbitrarily and add these groups to $\mathcal{Z}$. Thus, we obtain a partitioning $\mathcal{Z}$ of all centers into groups of LP weight $1/\beta$.

Analysis. We show that the algorithm returns a valid solution, and then prove an upper bound on its expected cost. For the sake of presentation, we defer some technical claims to Appendix B.

The algorithm picks exactly one center from each group, so it always picks βk centers. Hence, it always outputs a valid solution. Let S be the set of centers output by the algorithm. Denote the radius of the ball B_x by R_x^{β}. For every center x, we estimate the expected distance from x to the closest center c in the solution S i.e. $\mathbb{E}[d(x,S)]$. We show that $\mathbb{E}[d(x,S)] \leq \alpha(\beta)\, R_x$ for $\alpha(\beta)$ as in equation (8). Since $LP = \sum_x R_x$, we conclude that the algorithm has an approximation factor of $\alpha(\beta)$.

Fix $x \in X$. Recall, that $C_x = \{c : z_{xc} > 0\}$ is the set of all centers that serve x in the LP solution. We upper bound $d(x,S)$ by $d(x, (C_x \cup B_{w(x)}) \cap S)$, which is the distance to the closest center in $C_x \cup B_{w(x)}$ chosen by the algorithm. Note that the solution S always contains at least one center in $B_{w(x)}$, so $(C_x \cup B_{w(x)}) \cap S \neq \varnothing$. For the proof, we pick a particular (random) center $f(x) \in (C_x \cup B_{w(x)}) \cap S$.

We define $f(x)$ using the following randomized procedure. Consider the partitioning $\mathcal{Z}$ of all centers into groups of measure $1/\beta$ used by the algorithm. Let $\widetilde{\mathcal{Z}} = \{Z \cap C_x : Z \in \mathcal{Z};\ Z \cap C_x \neq \varnothing\}$ be the induced partitioning of the set C_x. For all $\widetilde{Z} \in \widetilde{\mathcal{Z}}$ we independently flip a coin and with probability $(1 - e^{-\beta y(\widetilde{Z})})/(\beta y(\widetilde{Z}))$ make the set $\widetilde{Z}$ *active*. We let $A \subset C_x$ to be the union of all active sets $\widetilde{Z}$; we say that centers in A are active centers. Let $f(x)$ be the center in $A \cap S$ closest to x, if $A \cap S \neq \varnothing$; let $f(x)$ to be the unique center in $B_{w(x)} \cap S$, otherwise. We set $\mathcal{E} = 0$, if $A \cap S \neq \varnothing$; and $\mathcal{E} = 1$, otherwise. Roughly speaking, $\mathcal{E}$ indicates whether $f(x) \in C_x$ or $f(x) \in B_{w(x)}$: Specifically, if $\mathcal{E} = 0$, then $f(x) \in C_x$; if $\mathcal{E} = 1$, then $f(x) \in B_{w(x)}$. Note, however, that $C_x \cap B_{w(x)} \neq \varnothing$, and $f(x)$ may belong to $C_x \cap B_{w(x)}$.

The center $f(x)$ may not be the closest to x, but since $f(x) \in S$, we have

$$d(x, S) \leq d(x, (C_x \cup B_{w(x)}) \cap S) \leq d(x, f(x)).$$

Let us now derive bound on the expected distance of a single client x to $f(x)$. We begin by considering the probability of the event $\mathcal{E}$.

▶ **Lemma 9.** $\Pr(\mathcal{E} = 0) = 1 - e^{-\beta}$.

Proof. Recall that the algorithm picks one center c in every $Z \in \mathcal{Z}$ uniformly (with respect to the measure y) at random. Thus, the probability that the algorithm picks a center from $\tilde{Z}$ equals $\beta y(\tilde{Z})$. The probability that a given $\tilde{Z}$ contains a point from the solution S and $\tilde{Z}$ is active equals $\beta y(\widetilde{Z}) \times (1 - e^{\beta y(\widetilde{Z})})/(\beta y(\widetilde{Z})) = (1 - e^{\beta y(\widetilde{Z})})$. The probability that no such $\widetilde{Z}$ exists equals

$$\prod_{\widetilde{Z} \in \widetilde{\mathcal{Z}}} e^{-\beta y(\widetilde{Z})} = e^{-\sum_{\widetilde{Z} \in \widetilde{\mathcal{Z}}} \beta y(\widetilde{Z})} = e^{-\beta y(C_x)} = e^{-\beta}. \qquad ◀$$

Using Lemma 9 we have:

$$\begin{aligned} E[d(x, f(x))] &= \Pr(\mathcal{E} = 0)\, \mathbb{E}\big[d(x, f(x)) \mid \mathcal{E} = 0\big] + \Pr(\mathcal{E} = 1)\, \mathbb{E}\big[d(x, f(x)) \mid \mathcal{E} = 1\big] \\ &= (1 - e^{-\beta})\, \mathbb{E}\big[d(x, f(x)) \mid \mathcal{E} = 0\big] + e^{-\beta}\, \mathbb{E}\big[d(x, f(x)) \mid \mathcal{E} = 1\big]. \end{aligned} \tag{9}$$

Let us now bound each remaining above, in turn.

▶ **Lemma 10.** $E[d(x, f(x)) \mid \mathcal{E} = 0] \leq R_x$.

Proof. We define two sets of random variables P and Q, and then show that they are identically distributed. If the algorithm picks a center c in $\widetilde{Z}$, and $\widetilde{Z}$ is active, let $P(\widetilde{Z}) = c$. Let $P(\widetilde{Z}) = \perp$, otherwise. The random variables $P(\widetilde{Z})$ are mutually independent for all $\widetilde{Z} \in \widetilde{\mathcal{Z}}$; and

$$\Pr(\widetilde{Z} = c) = \frac{(1 - e^{-\beta y(\widetilde{Z})})\, y_c}{y(\widetilde{Z})}$$

for $c \in \widetilde{Z}$.

To define Q, we introduce an auxiliary Poisson arrival process. At every point of time $t \in [0, \beta]$, we pick a center $c \in C_x$ with probability $y_c dt$ (i.e., with arrival rate y_c). For every $\widetilde{Z}$, let $Q(\widetilde{Z})$ be the first center chosen in $\widetilde{Z}$. If no centers in $\widetilde{Z}$ are chosen, we let $Q(\widetilde{Z}) = \perp$. Note that we pick two centers at exactly the same time with probability 0, hence $Q(\widetilde{Z})$ is well defined. Conditional on $Q(\widetilde{Z}) \neq \perp$, the random variable $Q(\widetilde{Z})$ is uniformly distributed in $\widetilde{Z}$ with respect to the LP measure y (since at every given time t, the probability of arrival equals $y_c dt$). Then, $\Pr(Q(\widetilde{Z}) \neq \perp) = (1 - e^{-\beta y(\widetilde{Z})})$. Hence, $\Pr(Q(\widetilde{Z}) = c) = (1 - e^{-\beta y(\widetilde{Z})}) y_c / y(\widetilde{Z})$. Note that all random variables Q are mutually independent. Thus, the random variables Q have the same distribution as random variables P.

Note that if $\mathcal{E} = 0$, then $f(x)$ is the closest center in $\{P(\widetilde{Z}) : \widetilde{Z} \in \widetilde{\mathcal{Z}}; P(\widetilde{Z}) \neq \perp\}$ to x. If $\mathcal{E} = 1$, then all $P(\widetilde{Z})$ are equal to $\perp$. Let $U_Q = \{Q(\widetilde{Z}) : \widetilde{Z} \in \widetilde{\mathcal{Z}}; Q(\widetilde{Z}) \neq \perp\}$. Since P and Q have the same distribution, we have

$$\mathbb{E}[d(x, f(x)) \mid \mathcal{E} = 0] = \mathbb{E}[\min_{c \in U_Q} d(x, c) \mid U_Q \neq \varnothing].$$

Conditional on $U_Q \neq \varnothing$, the first center that arrives according to our stochastic process is uniformly distributed in C_x, according the measure y. The expected distance from it to x equals R_x. Hence, $\mathbb{E}[\min_{c \in U_Q} d(x, c) \mid U_Q \neq \varnothing] \leq R_x$. ◀

Observe that for a random center c distributed according to the LP measure y in C_x (i.e., $\Pr(c = c_0) = y(c_0)/y(C_x) = y(c_0)$), we have the exact equality $\mathbb{E}[d(x,c)] = R_x$. So Lemma 10 shows that the distribution of $f(x)$ given $\mathcal{E} = 0$ is "not worse" than the distribution according to y in C_x.

We now bound the expected distance from x to $f(x)$ given $\mathcal{E} = 1$.

▶ **Lemma 11.** *Let $\gamma = \beta y(D_x)$. Then,*

$$\mathbb{E}[d(x,f(x)) \mid \mathcal{E} = 1] \leq \left(e^{\gamma}(1-\gamma) \times 3\Big(\frac{\beta}{\beta-1} + r_1 + r_2\Big) + (1 - e^{\gamma}(1-\gamma)) \times \frac{\beta}{\gamma}\right) R_x,$$

for some non-negative numbers r_1 and r_2 such that $r_1 \leq r_2$ and $\frac{1-\gamma}{\beta} r_1 + \frac{\beta-1}{\beta} r_2 \leq R_x$.

Proof. Recall, that $w(x)$ is the witness for x. Thus, the balls B_x and $B_{w(x)}$ intersect and $R_{w(x)} \leq R_x$. Let $c_\circ$ be an arbitrary center in $B_x \cap B_{w(x)}$. By the relaxed 3-hop triangle inequality,

$$\begin{aligned} d(x,f(x)) &\leq 3\big(d(x,c_\circ) + d(w(x),c_\circ) + d(w(x),f(x)\big) \\ &\leq 3\big(R_x^\beta + R_{w(x)}^\beta + d(w(x),f(x))\big). \end{aligned} \tag{10}$$

Here, we used that R_x^β is the radius of B_x; $R_{w(x)}^\beta$ is the radius of $B_{w(x)}$. Now, let $D_x = B_{w(x)} \cap C_x$. In Lemmas 15 and 16 in Appendix B, we show that:

$$R_x^\beta \leq \beta R_x/(\beta - 1) \tag{11}$$

and

$$R_{w(x)}^\beta + \mathbb{E}[d(w(x),f(x)) \mid f(x) \in B_{w(x)} \setminus D_x; \mathcal{E} = 1] \leq r_1 + r_2, \tag{12}$$

for some pair of nonnegative numbers r_1 and r_2 ($r_1 \leq r_2$) satisfying the conditions of the lemma. Taking expectations conditioned on $f(x) \in B_{w(x)} \setminus D_x$ and $\mathcal{E} = 1$ in (10), and then applying the bounds (11) and (12), we obtain:

$$\mathbb{E}[d(x,f(x)) \mid f(x) \in B_{w(x)} \setminus D_x; \mathcal{E} = 1] \leq 3\Big(\frac{\beta R_x}{\beta - 1} + r_1 + r_2\Big). \tag{13}$$

In Lemmas 17 and 18 in the Appendix, we show, respectively, that

$$\Pr(f(x) \in B_{w(x)} \setminus D_x \mid \mathcal{E} = 1) = e^{\gamma}(1-\gamma),$$

and

$$\mathbb{E}[d(x,f(x)) \mid f(x) \in D_x; \mathcal{E} = 1] \leq \frac{\beta R_x}{\gamma}.$$

Combining these bounds with (13), we obtain the desired inequality. ◀

Combining Lemmas 10 and 11 with (9), we have:

$$\mathbb{E}[d(x,f(x))] \leq R_x\left((1-e^{-\beta}) + e^{-\beta}\left(e^{\gamma}(1-\gamma) \times 3\Big(\frac{\beta}{\beta-1} + \frac{r_1+r_2}{R_x}\Big) + (1-e^{\gamma}(1-\gamma)) \times \frac{\beta}{\gamma}\right)\right).$$

Now, we recall that $\gamma = \beta y(D_x)$, and note that $\gamma \in [0,1]$, since $y(B_{w(x)}) = 1/\beta$. In order to bound the right hand side, we take the maximum over all $\gamma \in [0,1]$ and $r_1, r_2 \geq 0$

satisfying the conditions given in Lemma 11. The right hand side is a linear function of r_1 and r_2. Hence, for a fixed γ the maximum is attained at one of the two extreme points: $(r_1, r_2) = (0, \beta R_x/(\beta - 1))$ or $(r_1, r_2) = (\beta R_x/(\beta - \gamma), \beta R_x/(\beta - \gamma))$. Substituting r_1 and r_2 in the previous inequality we get the following upper bound on the ratio $\mathbb{E}[d(x, f(x))]/R_x$:

$$\max_{\gamma \in [0,1]} \Big((1-e^{-\beta}) + 3e^{-(\beta-\gamma)}(1-\gamma)\Big(\frac{\beta}{\beta-1} + \max\Big(\frac{\beta}{\beta-1}, \frac{2\beta}{\beta-\gamma}\Big)\Big) + \frac{\beta e^{-\beta}(1-e^{\gamma}(1-\gamma))}{\gamma}\Big). \tag{14}$$

This function can in turn be upper bounded by $\alpha(\beta)$, as defined in (8). We conclude that the approximation factor of the algorithm is upper bounded by $\alpha(\beta)$.

5 Local Search

For smaller values of β, we consider the standard local search algorithm (see, e.g., [6]) for the k-median problem using swaps of size s, but now allow the solution to contain βk centers. The algorithm works as follows: we maintain a current solution A comprising βk centers in $\mathcal{C}$. We repeatedly attempt to reduce the cost of the current solution A by closing a set of at most s centers in A and opening the same number of new centers from $\mathcal{C} \setminus A$. When no such local swap improves the cost of the solution A we terminate and return A. In order to simplify our analysis, we do not worry about convergence time of the algorithm here. We note that by applying standard techniques (see [6, 9]), we can ensure that, for any $\delta > 0$, the algorithm converges in time polynomial in $n = |\mathcal{C} \cup \mathcal{D}|$ and $\frac{1}{\delta}$ by instead stopping when no local swap improves the cost of A by a factor of $\left(1 - \frac{\delta}{\mathrm{poly}(n)}\right)$; the resulting algorithm's approximation ratio increases by only $\frac{1}{1-\delta}$.

Unfortunately standard analyses of local search algorithms for the k-median problem[6, 9] rely heavily on the triangle inequality, while the instances generated by Theorem 7 satisfy only a 3-relaxed 3-hop triangle inequality. Thus, we proceed as in Kanungo et al. [18]. Similarly to the previous section, we defer some technical details to Appendix C.

Let $O = \langle o_1, \ldots, o_k \rangle$ be an optimal set of k centers, and $A = \langle a_1, \ldots, a_{\beta k} \rangle$ be the set of βk centers produced by the local search algorithm. As in [18], we say that a center $a \in A$ *captures* a center $o \in O$ if a is the center of A that is closest to o. Note that each center in A can potentially capture several centers in O, but each center in O is captured by exactly one center of A. We now construct a set of local swaps to consider in our analysis. We say that a center in A is "good" if it does not capture any center of O. Then, because each center of O is captured by only one center of A, we must have at least $\beta k - k = (\beta - 1)k$ good centers in A. We fix some such set of $(\beta - 1)k$ good centers; we call them "auxiliary" centers and set them aside for now.

For the remaining k centers $B \subseteq A$, we proceed exactly as in [18]: we assign each center in O to the bad center of B that captures it. This creates a partition $O_1, \ldots, O_r$ of centers in O. We similarly partition the centers of B into r parts $B_1, \ldots, B_r$ with $|B_i| = |O_i|$; for each $1 \le i \le r$, let B_i contain the bad center of B that captures all of O_i together with $|B_i| - 1$ unique good centers of B. Note that the fact that each center of O is captured only once ensures that there are indeed enough good centers in B for our construction. Now, we use this partition of B and O to construct a set of swaps, each assigned some weight. If $|O_i| \le s$, we consider the $\langle B_i, O_i \rangle$ with weight 1. If $|O_i| = q > s$, we consider the group of all singleton swaps $\langle \{b\}, \{o\} \rangle$, where $o \in O_i$ and b is a good center in B_i, each given weight $\frac{1}{q-1}$. At this point, note that every center in O occurs in swaps of total weight 1, and every center in B

occurs in swaps of total weight at most $\frac{q}{q-1} \leq 1 + \frac{1}{s}$. Now, we add swaps involving auxiliary centers; for each of the $(\beta - 1)k$ auxiliary centers $a \in A \setminus B$ and each $o \in O$, we consider singleton swap $\langle \{a\}, \{o\} \rangle$, assigned weight $\frac{1}{k}$. Each center of O now occurs in swaps of total weight $1 + (\beta - 1) = \beta$, while each center of $A \setminus B$ occurs in swaps of total weight 1.

Summarizing, our set of swaps satisfies the following properties: (1) each center of O occurs in swaps of total weight β; (2) each center of A occurs in swaps of total weight at most $1 + \frac{1}{s}$; (3) for any swap $\langle A', O' \rangle$ in our set, no center in A' captures any center not in O'. We now give a brief sketch of how these properties lead to our desired approximation ratio (we give a full description of the analysis in the appendix). Our analysis closely follows that of [18].

Recall that for some set C of centers and some $c \in C$, we denote by $N_C(c)$ the set of all points x whose closest center in C is c. As in [18], the total change $\mathrm{cost}_{\mathcal{D},d}(A \setminus A' \cup O') - \mathrm{cost}_{\mathcal{D},d}(A)$ due to performing a single swap $\langle A', O' \rangle$ is at most:

$$\sum_{o \in O'} \sum_{x \in N_O(o)} \big(d(x, o) - d(x, a_x)\big) + \sum_{a \in A'} \sum_{x \in N_A(A')} \big(d(x, a_{o_x}) - d(x, a_x)\big).$$

If A is locally optimal, then we must have that $\mathrm{cost}_{\mathcal{D},d}(A \setminus A' \cup O') - \mathrm{cost}_{\mathcal{D},d}(A) \geq 0$ for all swaps (A', O') considered by the algorithm. In particular, for each swap $\langle A', O' \rangle$ in our set, we have:

$$0 \leq \sum_{o \in O'} \sum_{x \in N_O(o)} \big(d(x, o) - d(x, a_x)\big) + \sum_{a \in A'} \sum_{x \in N_A(A')} \big(d(x, a_{o_x}) - d(x, a_x)\big). \tag{15}$$

Multiplying each inequality (15) by the weight of its swap and then adding the resulting inequalities we obtain:

$$0 \leq \beta \sum_{x \in \mathcal{D}} (d(x, o_x) - d(x, a_x)) + \left(1 + \frac{1}{s}\right) \sum_{x \in \mathcal{D}} (d(x, a_{o_x}) - d(x, a_x)),$$

due to properties (1) and (2) of our set of swaps. Theorem 7 part 2, which shows that our center set is an approximate k-means centroid set, then allows us to simplify the final term above as in [18], giving:

$$0 \leq \left(\beta + 2 + \tfrac{2}{s}\right) \mathrm{cost}_{\mathcal{D},d}(O) - \left(\beta - \frac{2 + \frac{2}{s}}{\alpha}\right) \mathrm{cost}_{\mathcal{D},d}(A) + O(\epsilon) \cdot \mathrm{cost}_{\mathcal{D},d}(A),$$

where $\alpha^2 = \frac{\mathrm{cost}_{\mathcal{D},d}(A)}{\mathrm{cost}_{\mathcal{D},d}(O)}$ is the squared approximation ratio of our algorithm. Rearranging and simplifying (again, we give a detailed analysis in the appendix), we obtain

$$\alpha < \left(1 + \frac{2}{\beta} + \frac{2}{\beta s}\right) \frac{1}{1 - O(\epsilon)}.$$

Thus, we have the following theorem:

▶ **Theorem 12.** *There exists an algorithm that produces a solution for any instance of βk-median problem satisfying the properties of Theorem 7, where $\beta > 1$ is a fixed constant. For any $s \geq 1$ and any $\varepsilon \in (0, 1]$, the algorithm runs in time polynomial in $|\mathcal{C} \cup \mathcal{D}|$ and produces a solution A satisfying:*

$$\mathrm{cost}_{\mathcal{D},d}(A) \leq \left(1 + \frac{2}{\beta} + \frac{2}{\beta s}\right)^2 \frac{1}{1 - O(\epsilon)} \cdot \mathrm{cost}_{\mathcal{D},d}(O)$$

where O is the optimal set of k centers in $\mathcal{C}$.

References

1 Alexander A. Ageev and Maxim Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *J. of Combinatorial Optimization*, 8(3):307–328, 2004.

2 Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *Proc. of the 12th International Workshop APPROX*, pages 15–28, 2009.

3 D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. In *Machine Learning*, volume 75, pages 245–249, 2009.

4 Aris Anagnostopoulos, Anirban Dasgupta, and Ravi Kumar. A constant-factor approximation algorithm for co-clustering. In *Theory of Computing*, volume 8(1), pages 597–622, 2012.

5 D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007.

6 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. on Computing*, 33(3):544–562, 2004.

7 Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of Euclidean k-means. In *Proc. of the 31st International Symposium on Computational Geometry*, pages 754–767, 2015.

8 Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proc. of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 737–756, 2015. Updated version: `http://arxiv.org/abs/1406.2951`.

9 Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM J. on Computing*, 34(4):803–824, 2005.

10 Fabian A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. on Computing*, 33(1):1–25, 2003.

11 Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proc. of the 47th Annual ACM on Symposium on Theory of Computing*, pages 163–172, 2015.

12 Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. *CoRR*, abs/1603.09535, 2016. URL: `http://arxiv.org/abs/1603.09535`.

13 S. Dasgupta. The hardness of k-means clustering. In *Technical Report CS2007-0890, University of California, San Diego.*, 2007.

14 Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A PTAS for k-means clustering based on weak coresets. In *Proc. of the 23rd Annual Symposium on Computational Geometry*, pages 11–18, 2007.

15 Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local search yields a PTAS for k-means in doubling metrics. *CoRR*, abs/1603.08976, 2016. URL: `http://arxiv.org/abs/1603.08976`.

16 Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering: (extended abstract). In *Proc. of the 10th Annual Symposium on Computational Geometry*, pages 332–339, 1994.

17 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM*, 48(2):274–296, March 2001.

18 Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Angela Y. Wu, and Christine D. Piatko. A local search approximation algorithm for k-means clustering. *Computational Geometry*, 28(2–3):89–112, 2004.

19 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proc. of the 45th Annual ACM Symposium on Theory of Computing*, pages 901–910, 2013.

20 Jyh-Han Lin and Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Lett.*, 44(5):245–249, December 1992.

21 Jyh-Han Lin and Jeffrey Scott Vitter. ϵ-approximations with minimum packing constraint violation. In *Proc. of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.

22 S. Lloyd. Least squares quantization in PCM. Technical report, Bell Laboratories, 1957.

23 S. Lloyd. Least squares quantization in PCM. *IEEE Trans. on Information Theory*, 28(2):129–137, Mar 1982.

24 Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is NP-hard. In *Proc. of the 3rd International Workshop on Algorithms and Computation*, pages 274–285, 2009.

25 Jirı Matoušek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84, 2000.

26 R. Ostrovsky, Y. Rabani, L. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. In *Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 165–174, 2006.

27 Maxim Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Integer Programming and Combinatorial Optimization: Proc. of the 9th IPCO Conference*, pages 240–257, 2002.

A Proof of Theorem 7

In this section, we prove Theorem 7. Consider an instance of k-means with a set of points $X \subset \mathbb{R}^p$. Denote $n = |X|$. Let $\varepsilon' = \varepsilon/3$. Let $\varphi : \mathbb{R}^p \to \mathbb{R}^{\tilde{p}}$ be a dimension reduction transform for X with distortion $(1+\varepsilon')$ as in Theorem 6. Note that $\tilde{p} = O(\log n/\varepsilon'^2) = O(\log n/\varepsilon^2)$.

Let $X' = \varphi(X) \subset \mathbb{R}^{\tilde{p}}$. Using the algorithm from Theorem 5, we compute an ε'-approximate centroid set $\mathcal{C} \subset \mathbb{R}^{\tilde{p}}$ for X'. The size of $\mathcal{C}$ is

$$O(n\varepsilon^{-\tilde{p}}\log(1/\varepsilon)) = n\varepsilon^{-O(\log n/\varepsilon^2)}\log(1/\varepsilon) = n \cdot n^{O(\log(1/\varepsilon)/\varepsilon^2)} = n^{O(\log(1/\varepsilon)/\varepsilon^2)};$$

we need time $O(n\log n + n\varepsilon^{-\tilde{p}}\log(1/\varepsilon)) = n^{O(\log(1/\varepsilon)/\varepsilon^2)}$ to compute it.

We first show that for every solution of the k-means problem on X there is a corresponding solution of k-means problem on X' in which all centers lie in $\mathcal{C}$, and vice versa.

▶ **Lemma 13.** *The following hold:*

1. *For every partition $S = \langle S_1, \dots, S_k\rangle$ of X, there is a corresponding clustering of X' given by $S' = \langle\varphi(S_1), \dots, \varphi(S_k)\rangle$ and some centers $C' = \langle c'_1, \dots, c'_k\rangle \subseteq \mathcal{C}$ such that:*

$$\operatorname{cost}_{X'}(S', C') \leq (1+\varepsilon')^2 \operatorname{cost}_X(S).$$

2. *For every partition $S' = \langle S'_1, \dots, S'_k\rangle$ of X', there is a corresponding clustering $S = \langle\varphi^{-1}(S_1), \dots, \varphi^{-1}(S_k)\rangle$ of X and some centers $C = \langle c_1, \dots, c_k\rangle \subseteq \mathbb{R}^p$ such that*

$$\operatorname{cost}_X(S, C) \leq \operatorname{cost}_{X'}(S').$$

Proof. Part 1: Consider a partition $S = \langle S_1, \dots, S_k\rangle$ of X and the corresponding partition $S' = \langle S'_1, \dots, S'_k\rangle$ of X', where $S'_i = \varphi(S_i)$. Let $c'_i = \arg\min_{c\in\mathcal{C}} \sum_{x\in S'_i} \|x' - c\|^2$ for

$i \in \{1, \dots, k\}$. Because $\mathcal{C}$ is an ε'-approximate centroid set for X', we have, for each cluster S'_i,

$$\sum_{x \in S'_i} \|x - c'_i\|^2 \le (1+\varepsilon') \min_{c \in \mathbb{R}^{\tilde{p}}} \sum_{x \in S'_i} \|x - c\|^2 = (1+\varepsilon') \frac{1}{2|S'_i|} \sum_{x', x'' \in S'_i} \|x' - x''\|^2$$
$$= \frac{1+\varepsilon'}{2|S'_i|} \sum_{x', x'' \in S_i} \|\varphi(x') - \varphi(x'')\|^2 \le \frac{(1+\varepsilon')^2}{2|S_i|} \sum_{x', x'' \in S_i} \|x' - x''\|^2$$

Hence,

$$\mathrm{cost}_{X'}(S') = \sum_{i=1}^{k} \sum_{x \in S'_i} \|x - c'_i\|^2 \le (1+\varepsilon')^2 \,\mathrm{cost}_X(S).$$

Part 2: Consider a partition $S' = \langle S'_1, \dots, S'_k \rangle$ of X' and the corresponding partition $S = \langle S_1, \dots, S_k \rangle$ of X, where $S_i = \varphi^{-1}(S'_i)$. Define the centers $c_i = \sum_{x \in S_i} x/|S_i|$. Then, for each cluster S_i, we have:

$$\sum_{x \in S_i} \|x - c_i\|^2 = \frac{1}{2|S_i|} \sum_{x', x'' \in S_i} \|x' - x''\|^2 \le \frac{1}{2|S_i|} \sum_{x', x'' \in S_i} \|\varphi(x') - \varphi(x'')\|^2$$
$$= \frac{1}{2|S'_i|} \sum_{x', x'' \in S'_i} \|x' - x''\|^2.$$

Hence,

$$\mathrm{cost}_X(S) = \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|^2 \le \mathrm{cost}_{X'}(S'). \qquad ◀$$

Now we are ready to define instance $\langle \mathcal{D}, \mathcal{C}, d \rangle$. Let $\mathcal{D} = X'$, $\mathcal{C}$ be the ε-approximate centroid we defined above, and $d(c, x) = \|c - x\|^2$ for every $c \in \mathcal{C}$ and $x \in \mathcal{D}$. Define $\psi : \mathcal{D} \to X$ by $\psi(x) = \varphi^{-1}(x)$.

We prove that our reduction, which maps instance X of k-means to instance $\langle \mathcal{D}, \mathcal{C}, d \rangle$ of k-median, satisfies the conditions of the theorem.

▶ **Lemma 14.** *Our reduction produces an instance that satisfies the following properties:*

1. *The distance function d satisfies the 3-relaxed 3-hop triangle inequality on $\mathcal{D} \cup \mathcal{C}$.*
2. *For every partition $S = \langle S_1, \dots, S_k \rangle$ of $\mathcal{D}$ and the corresponding partition $\psi(S) = \langle \psi(S_1), \dots, \psi(S_k) \rangle$ of X, we have*
$$\mathrm{cost}_X(\psi(S)) \le \mathrm{cost}_{\mathcal{D},d}(S) \le (1+\varepsilon)\,\mathrm{cost}_X(\psi(S)).$$
3. *We have*
$$\mathrm{OPT}_X \le \mathrm{OPT}_{\langle \mathcal{D}, d \rangle} \le (1+\varepsilon)\mathrm{OPT}_X.$$

Proof. Claim 1 follows from the fact that:

$$\|x - w\|^2 \le (\|x - y\| + \|y - z\| + \|z - w\|)^2 \le 3(\|x - y\|^2 + \|y - z\|^2 + \|z - w\|^2).$$

for any $w, x, y, z \in \mathbb{R}^{\tilde{p}}$.

For claim 2, consider any partition S of $\mathcal{D}$. Let $T = \psi(S)$ be the corresponding partition of X, given by $T_i = \psi(S_i)$. Then, from our definition of d, we have $\mathrm{cost}_{\mathcal{D},d}(S) = \mathrm{cost}_{X'}(S)$.

Moreover, by Lemma 13, we have $\mathrm{cost}_{X'}(S)$ is between $\mathrm{cost}_X(T)$ and $(1+\varepsilon')^2\,\mathrm{cost}_X(T)$. Thus,

$$\mathrm{cost}_X(\psi(S)) \le \mathrm{cost}_{\mathcal{D},d}(S) \le (1+\varepsilon')^2\,\mathrm{cost}_X(\psi(S)) \le (1+\varepsilon)\,\mathrm{cost}_X(\psi(S)).$$

Since for every partition S of $\mathcal{C}$ there is a corresponding partition $\psi(S)$ of X, and for every partition T of X there is a corresponding partition $\varphi(T)$ of $\mathcal{D}$, we immediately get from claim 2 that $\mathrm{OPT}_X \le \mathrm{OPT}_{\langle\mathcal{D},d\rangle} \le (1+\varepsilon)\mathrm{OPT}_X$. ◀

B Detailed Analysis of the LP Rounding Algorithm

Here we give a detailed proof of the necessary facts for the analysis of Section 4.

▶ **Lemma 15.** *The following inequality holds: $R_x^\beta \le \beta R_x/(\beta-1)$.*

Proof. We have

$$R_x = \sum_{c\in C_x} y_c d(x,c) \le \sum_{c\in C_x\setminus B_x} y_c d(x,c).$$

Every center $c \in C_x \setminus B_x$ is at distance at least R_x^β from x. Hence,

$$R_x \le \sum_{c\in C_x\setminus B_x} y_c R_x^\beta = y(C_x\setminus B_x)\,R_x^\beta = \left(1-\frac{1}{\beta}\right)R_x^\beta.$$

The desired inequality follows. ◀

▶ **Lemma 16.** *There exist two nonnegative numbers r_1 and r_2 satisfying*

1. $\left(\frac{1-\gamma}{\beta}\right)r_1 + \left(\frac{\beta-1}{\beta}\right)r_2 \le R_x$,

2. $r_1 \le r_2$,

such that $R_{w(x)}^\beta + \mathbb{E}[d(w(x),f(x)) \mid f(x)\in B_{w(x)}\setminus D_x;\ \mathcal{E}=1] \le r_1 + r_2$.

Proof. Denote the expected distance from a random center c in $B_{w(x)}\setminus D_x$ to $w(x)$ by r_1 and distance from a random center c in $C_{w(x)}\setminus B_{w(x)}$ to $w(x)$ by r_2:

$$r_1 = \frac{\sum_{c\in B_{w(x)}\setminus D_x} y_c d(w(x),c)}{y(B_{w(x)}\setminus D_x)} \qquad\qquad r_2 = \frac{\sum_{c\in C_{w(x)}\setminus B_{w(x)}} y_c d(w(x),c)}{y(C_{w(x)}\setminus B_{w(x)})}.$$

By the definition of $R_{w(x)}$, we have

$$\begin{aligned} R_{w(x)} &= \Big(\sum_{c\in D_x} y_c d(w(x),c)\Big) + y(B_{w(x)}\setminus D_x)\,r_1 + y(C_{w(x)}\setminus B_x)\,r_2 \\ &\ge \left(\frac{1-\gamma}{\beta}\right)r_1 + \left(\frac{\beta-1}{\beta}\right)r_2, \end{aligned}$$

since $y(B_x) = y(B_{w(x)}) = 1/\beta$, $y(C_{w(x)}) = 1$, and $y(D_x) = \gamma/\beta$. Note that $R_{w(x)} \le R_x$. Hence,

$$\left(\frac{1-\gamma}{\beta}\right)r_1 + \left(\frac{\beta-1}{\beta}\right)r_2 \le R_x.$$

Since all centers in $B_{w(x)}\setminus D_x$ lie inside of the ball of radius $R_{w(x)}^\beta$ around $w(x)$, and all centers in $C_{w(x)}\setminus B_{w(x)}$ lie outside of this ball, we have $r_1 \le R_{w(x)}^\beta \le r_2$. Hence,

$R^{\beta}_{w(x)} + d(w(x), f(x)) \le r_2 + d(w(x), f(x))$ and $r_1 \le r_2$. Conditional on $f(x) \in B_{w(x)} \setminus D_x$ and $\mathcal{E} = 1$, the random center $f(x)$ is distributed uniformly in $B_{w(x)} \setminus D_x$ with respect to the LP measure y. Hence, $\mathbb{E}[d(w(x), f(x)) \mid f(x) \in B_{w(x)} \setminus D_x; \mathcal{E} = 1] = r_1$. Consequently,

$$R^{\beta}_{w(x)} + \mathbb{E}[d(w(x), f(x)) \mid f(x) \in B_{w(x)} \setminus D_x; \mathcal{E} = 1] \le r_1 + r_2. \qquad ◀$$

▶ **Lemma 17.** *We have* $\Pr(f(x) \in D_x \mid \mathcal{E} = 1) = 1 - e^{\gamma}(1 - \gamma)$.

Proof. Observe that the set $D_x = B_{w(x)} \cap C_x$ is one of the sets in the partitioning $\widetilde{\mathcal{Z}}$ as $w(x) \in \mathcal{W}$ and $B_{w(x)} \in \mathcal{Z}$. Assume $f(x) \in D_x$ and $\mathcal{E} = 1$. Since $f(x) \in D_x$, we have $S \cap D_x \ne \varnothing$. Thus, D_x must be inactive (otherwise, $\mathcal{E}$ would be 0). Moreover, for every $\widetilde{Z} \ne D_x$ ($\widetilde{Z} \in \mathcal{Z}$), $\widetilde{Z}$ is inactive or $\widetilde{Z} \cap S = \varnothing$ (again, otherwise, $\mathcal{E}$ would be 0). Hence, the event

$$\{f(x) \in D_x \text{ and } \mathcal{E} = 1\}$$

can be represented as the intersection of the following three independent events: $\{S \cap D_x \ne \varnothing\}$, $\{D_x$ is not active$\}$, and $\{$there are no active centers in $(C_x \setminus D_x) \cap S\}$. The probability of the first event is $\beta y(D_x)$; the probability of the second event is $1 - (1 - e^{-\beta y(D_x)})/(\beta y(D_x))$; the probability of the third event is $e^{-\beta y(C_x \setminus D_x)}$ (this probability is computed as in Lemma 9). Thus,

$$\begin{aligned}\Pr(f(x) \in D_x \text{ and } \mathcal{E} = 1) &= \beta y(D_x) \times \Big(1 - \frac{1 - e^{-\beta y(D_x)}}{\beta y(D_x)}\Big) \times e^{-\beta y(C_x \setminus D_x)} \\ &= \big(\gamma - (1 - e^{-\gamma})\big) \times e^{-(\beta-\gamma)} = e^{-\beta}\big(1 - (1-\gamma)e^{\gamma}\big).\end{aligned}$$

Combining this with Lemma 9, which shows that $\Pr(\mathcal{E} = 1) = e^{-\beta}$ completes the proof. ◀

▶ **Lemma 18.** *The following bound holds:* $\mathbb{E}[d(x, f(x)) \mid f(x) \in D_x; \mathcal{E} = 1] \le \frac{\beta R_x}{\gamma}$.

Proof. Given $f(x) \in D_x$ and $\mathcal{E} = 1$, the random center $f(x)$ is distributed uniformly in D_x with respect to the LP measure y. Hence, $\Pr(f(x) = c) = y_c / y(D_x)$ for $c \in D_x$. We have

$$\mathbb{E}[d(x, f(x)) \mid f(x) \in D_x; \mathcal{E} = 1] = \frac{\sum_{c \in D_x} y_c d(x, c)}{y(D_x)} \le \frac{\sum_{c \in C_x} y_c d(x, c)}{y(D_x)} = \frac{R_x}{\gamma/\beta}. \qquad ◀$$

C Detailed Analysis of the Local Search Algorithm

Here we give a detailed analysis of the local search algorithm from Section 5, closely following Kanungo et al. [18].

For a set of points $P \subseteq X$ and a point $c \in \mathbb{R}^p$, define the total distortion of P with respect to c as $\Delta(P, c) \equiv \sum_{c' \in P} \|c' - c\|^2$. We shall use the following Lemmas from [18]:

▶ **Lemma 19** (Lemma 2.1 in [18]). *Given a finite subset P of points in $\mathbb{R}^p$, let c be the centroid of P. Then, for any $c' \in \mathbb{R}^p$,* $\Delta(P, c') = \Delta(P, c) + |P| \cdot \|c - c'\|^2$

▶ **Lemma 20.** *Let $\langle \rho_i \rangle$ and $\langle \xi_i \rangle$ be two sequences of reals such that* $\alpha^2 = (\sum_i \rho_i^2)/(\sum_i \xi_i^2)$ *for some $\alpha > 0$. Then,*

$$\sum_{i=1}^{n} \rho_i \xi_i \le \frac{1}{\alpha} \sum_{i=1}^{n} \rho_i^2.$$

We now show how local optimality implies the desired inequality. For a point $x \in \mathcal{D}$, let a_x and o_x denote the closest facility to x in A and O, respectively. Recall that for for $a \in A$, $N_A(a)$ is precisely the set of all those points $x \in \mathcal{D}$ such that $a_x = a$, and, similarly, for $o \in O$, $N_O(o)$ is the set of all points $x \in \mathcal{D}$ such that $o_x = o$. Now, we upper bound the change in cost due to some swap $\langle A', O' \rangle$ in our set of swaps. We do this by constructing a feasible assignment of all points in $\mathcal{D}$ to centers in $A \setminus A' \cup O'$. For each $o \in O'$, we assign all the points in $N_O(o)$ to o. This changes the cost by

$$\sum_{o \in O'} \sum_{x \in N_O(o)} (d(x, o) - d(x, a_x)).$$

Now, fix a point $x \in N_A(A') \setminus N_O(O')$, and consider x's closest optimal center o_x. We must have $o_x \notin O'$. Let a_{o_x} be the closest center to o_x in A. Then, by property (3) of our set of swaps, $a_{o_x} \notin A'$, since a_{o_x} captures o_x but $o_x \notin O'$. We reassign x to a_{o_x}. The total cost of reassigning all such points x is at most:

$$\sum_{a \in A'} \sum_{x \in N_A(A') \setminus N_O(O')} (d(x, a_{o_x}) - d(x, a_x)) \leq \sum_{a \in A'} \sum_{x \in N_A(A')} (d(x, a_{o_x}) - d(x, a_x)),$$

where the inequality follows from the fact that a_x is the closest center to x in A, and so $d(x, a_{o_x}) - d(x, a_x) \geq 0$ for all $x \in N_A(A') \cap N_O(O')$. Thus, the total change $\mathrm{cost}_{\mathcal{D},d}(A \setminus A' \cup O') - \mathrm{cost}_{\mathcal{D},d}(A)$ for each swap $\langle A', O' \rangle$ is at most:

$$\sum_{o \in O'} \sum_{x \in N_O(o)} (d(x, o) - d(x, a_x) + \sum_{a \in A'} \sum_{x \in N_A(A')} (d(x, a_{o_x}) - d(x, a_x)).$$

If A is locally optimal, then we must have that $\mathrm{cost}_{\mathcal{D},d}(A \setminus A' \cup O') - \mathrm{cost}_{\mathcal{D},d}(A) \geq 0$ for all swaps (A', O') considered by the algorithm. In particular, for each swap $\langle A', O' \rangle$ in our set, we have:

$$0 \leq \sum_{o \in O'} \sum_{x \in N_O(o)} \big(d(x, o) - d(x, a_x)\big) + \sum_{a \in A'} \sum_{x \in N_A(A')} \big(d(x, a_{o_x}) - d(x, a_x)\big).$$

Set $\gamma = 1 + \frac{1}{p}$. Then, multiplying each such inequality by the weight of its swap and then adding the resulting inequalities we obtain

$$\begin{aligned} 0 &\leq \beta \sum_{x \in \mathcal{D}} \big(d(x, o_x) - d(x, a_x)\big) + \gamma \sum_{x \in \mathcal{D}} \big(d(x, a_{o_x}) - d(x, a_x)\big) \\ &= \beta \, \mathrm{cost}_{\mathcal{D},d}(O) - (\beta + \gamma) \, \mathrm{cost}_{\mathcal{D},d}(A) + \gamma \sum_{x \in \mathcal{D}} d(x, a_{o_x}), \end{aligned} \tag{16}$$

where we have exploited properties (1) and (2) of our set of swaps to bound the number of times a given center in O or A is counted in our sum of inequalities.

It remains to bound the final term in (16). Consider some $o \in O$, and let c be the centroid of $N_O(o)$. As above, we will let a_o denote the closest center in A to O. Then, note that:

$$\begin{aligned}
\Delta(N_O(o), a_o) &= \Delta(N_O(o), c) + |N_O(o)| \cdot \|c - a_o\|^2 && \text{(Lemma 19)}\\
&\le \Delta(N_O(o), o) + |N_O(o)| \cdot \|c - a_o\|^2 && \text{(c is the centroid of $N_O(o)$)}\\
&\le \sum_{x \in N_O(o)} \left[d(x,o) + (1+\varepsilon)\|o - a_o\|^2\right]\\
&\qquad \text{(Theorem 7 part 2, and the fact that o is an optimal center for $N_O(o)$)}\\
&\le \sum_{x \in N_O(o)} \left[d(x,o) + (1+\varepsilon)\|o - a_x\|^2\right].\\
&&& \text{(a_o is the closest center to o in A)}\\
&\le (1+\varepsilon) \sum_{x \in N_O(o)} \left[d(x,o) + \|o - a_x\|^2\right].
\end{aligned}$$

Let $\alpha^2 = \frac{\mathrm{cost}_{\mathcal{D},d}(A)}{\mathrm{cost}_{\mathcal{D},d}(O)} = \frac{\sum_{x\in\mathcal{D}} d(x,a_x)}{\sum_{x\in\mathcal{D}} d(x,o_x)}$ be the approximation ratio attained by the algorithm. Summing over all $o \in O$, and recalling that for all $x \in N_O(o)$ we have $o_x = o$, we obtain:

$$\begin{aligned}
\sum_{x\in\mathcal{D}} d(x, a_{o_x}) &= \sum_{o \in O} \Delta(N_O(o), a_o)\\
&\le (1+\varepsilon) \sum_{o\in O} \sum_{x \in N_O(o)} \left[d(x,o) + \|o - a_x\|^2\right]\\
&= (1+\varepsilon) \sum_{x\in\mathcal{D}} \left[d(x,o_x) + \|o_x - a_x\|^2\right]\\
&\le (1+\varepsilon) \sum_{x\in\mathcal{D}} \left[d(x,o_x) + \|x - o_x\|^2 + \|x - a_x\|^2 + 2\|x - o_x\|\|x - a_x\|\right]\\
&= (1+\varepsilon) \sum_{x\in\mathcal{D}} \left[2d(x,o_x) + d(x,a_x) + 2\|x - o_x\|\|x - a_x\|\right]\\
&\le (1+\varepsilon) \sum_{x\in\mathcal{D}} \left[2d(x,o_x) + d(x,a_x) + \frac{2}{\alpha} d(x,a_x)\right]\\
&= (1+\varepsilon)\left[2\,\mathrm{cost}_{\mathcal{D},d}(O) + \left(1 + \frac{2}{\alpha}\right)\mathrm{cost}_{\mathcal{D},d}(A)\right]. \qquad (17)
\end{aligned}$$

Where in the last inequality, we have applied Lemma 20 to the sequences ρ_i and ξ_i defined by:

$$\alpha^2 = \frac{\sum_{x\in\mathcal{D}} d(x,a_x)}{\sum_{x\in\mathcal{D}} d(x,o_x)} = \frac{\sum_{i=1}^n \rho_i}{\sum_{i=1}^n \xi_i}.$$

Applying the upper bound (17) to the final term of (16), we obtain:

$$\begin{aligned}
0 &\le \beta\,\mathrm{cost}_{\mathcal{D},d}(O) - (\beta+\gamma)\,\mathrm{cost}_{\mathcal{D},d}(A) + \gamma(1+\varepsilon)\left[2\,\mathrm{cost}_{\mathcal{D},d}(O) + \left(1+\tfrac{2}{\alpha}\right)\mathrm{cost}_{\mathcal{D},d}(A)\right]\\
&\le (\beta+2\gamma)\,\mathrm{cost}_{\mathcal{D},d}(O) - \left(\beta - \tfrac{2\gamma}{\alpha}\right)\mathrm{cost}_{\mathcal{D},d}(A) + \left(3 + \tfrac{2}{\alpha}\right)\gamma\varepsilon\,\mathrm{cost}_{\mathcal{D},d}(A)
\end{aligned}$$

where we have used the fact that $\mathrm{cost}_{\mathcal{D},d}(O) \le \mathrm{cost}_{\mathcal{D},d}(A)$. Rearranging, we have

$$\begin{aligned}
(\beta+2\gamma)\,\mathrm{cost}_{\mathcal{D},d}(O) &\ge \left(\beta - \frac{2\gamma}{\alpha} - \left(3+\tfrac{2}{\alpha}\right)\gamma\epsilon\right)\mathrm{cost}_{\mathcal{D},d}(A)\\
&= \left(\beta - \frac{2\gamma}{\alpha} - \left(3+\tfrac{2}{\alpha}\right)\gamma\epsilon\right)\alpha^2\,\mathrm{cost}_{\mathcal{D},d}(O),
\end{aligned}$$

which implies:

$$\begin{aligned}
\alpha^2\beta - 2\gamma\alpha - \beta - 2\gamma - \alpha^2\left(3 + \tfrac{2}{\alpha}\right)\gamma\epsilon &\le 0 \\
\alpha^2 - \frac{2\gamma\alpha}{\beta} - 1 - \frac{2\gamma}{\beta} - \frac{\alpha^2}{\beta}\left(3 + \tfrac{2}{\alpha}\right)\gamma\epsilon &\le 0 \\
(\alpha+1)\left(\alpha - 1 - \frac{2\gamma}{\beta}\right) - \frac{\alpha^2}{\beta}\left(3 + \tfrac{2}{\alpha}\right)\gamma\epsilon &\le 0 \\
\left(\alpha - 1 - \frac{2\gamma}{\beta}\right) - \frac{\alpha^2}{(\alpha+1)\beta}\left(3 + \tfrac{2}{\alpha}\right)\gamma\epsilon &\le 0.
\end{aligned}$$

Thus, we have:

$$(1 - O(\epsilon))\alpha \le 1 + \frac{2\gamma}{\beta} = 1 + \frac{2}{\beta} + \frac{2}{\beta p}.$$

Near-Optimal UGC-hardness of Approximating Max k-CSP$_R$*

Pasin Manurangsi[1], Preetum Nakkiran[2], and Luca Trevisan[3]

1 **University of California, Berkeley, USA**
`pasin@berkeley.edu`
2 **University of California, Berkeley, USA**
`preetum@berkeley.edu`
3 **University of California, Berkeley, USA**
`luca@berkeley.edu`

Abstract

In this paper, we prove an almost-optimal hardness for Max k-CSP$_R$ based on Khot's Unique Games Conjecture (UGC). In Max k-CSP$_R$, we are given a set of predicates each of which depends on exactly k variables. Each variable can take any value from $1, 2, \dots, R$. The goal is to find an assignment to variables that maximizes the number of satisfied predicates.

Assuming the Unique Games Conjecture, we show that it is NP-hard to approximate Max k-CSP$_R$ to within factor $2^{O(k \log k)}(\log R)^{k/2}/R^{k-1}$ for any k, R. To the best of our knowledge, this result improves on all the known hardness of approximation results when $3 \leq k = o(\log R/\log\log R)$. In this case, the previous best hardness result was NP-hardness of approximating within a factor $O(k/R^{k-2})$ by Chan. When $k = 2$, our result matches the best known UGC-hardness result of Khot, Kindler, Mossel and O'Donnell.

In addition, by extending an algorithm for Max 2-CSP$_R$ by Kindler, Kolla and Trevisan, we provide an $\Omega(\log R/R^{k-1})$-approximation algorithm for Max k-CSP$_R$. This algorithm implies that our inapproximability result is tight up to a factor of $2^{O(k \log k)}(\log R)^{k/2-1}$. In comparison, when $3 \leq k$ is a constant, the previously known gap was $O(R)$, which is significantly larger than our gap of $O(\operatorname{polylog} R)$.

Finally, we show that we can replace the Unique Games Conjecture assumption with Khot's d-to-1 Conjecture and still get asymptotically the same hardness of approximation.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases inapproximability, unique games conjecture, constraint satisfaction problem, invariance principle

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.15

1 Introduction

Maximum Constraint Satisfaction Problem (Max CSP) is an optimization problem where the inputs are a set of variables, an alphabet, and a set of predicates. Each variable can be assigned any symbol from the alphabet and each predicate depends only on the assignment to a subset of variables. The goal is to find an assignment to the variables that maximizes the number of satisfied predicates.

* This material is based upon work supported by the National Science Foundation under Grant No. CCF 1540685.

Many natural optimization problems, such as Max Cut, Max k-CUT and Max k-SAT, can be formulated as Max CSP. In addition, Max CSP has been shown to help approximate other seemingly-unrelated problems such as Densest k-Subgraph [4]. Due to this, Max CSP has long been researched by the approximation algorithm community [35, 18, 6, 26, 24, 14]. Furthermore, its relation to PCPs ensures that Max CSP is also well-studied on the hardness of approximation side [32, 11, 33, 22, 2, 16, 12, 3].

The main focus of this paper is on Max k-CSP$_R$, a family of Max CSP where the alphabet is of size R and each predicate depends on only k variables. On the hardness of approximation side, most early works focused on boolean Max k-CSP. Samorodnitsky and Trevisan first showed that Max k-CSP$_2$ is NP-hard to approximate to within factor $2^{O(\sqrt{k})}/2^k$ [32]. Engebretsen and Holmerin later improved constant factors in the exponent $O(\sqrt{k})$ but still yielded hardness of a factor $2^{O(\sqrt{k})}/2^k$ [12]. To break this barrier, Samorodnitsky and Trevisan proved a hardness of approximation conditioned on Khot's Unique Games Conjecture (UGC), which will be discussed in more detail later; they achieved a ratio of $O(k/2^k)$ hardness, which is tight up to a constant for the boolean case [33]. Chan later showed that NP-hardness of approximation at this ratio can be achieved unconditionally and, thus, settled down the approximability of Max k-CSP$_2$ [3].

Unlike the boolean case, the approximability of Max k-CSP$_R$ when $R > 2$ is still not resolved. In this case, Engebretsen showed $R^{O(\sqrt{k})}/R^k$ NP-hardness of approximation [11]. Under the Unique Games Conjecture, a hardness of approximation of $O(kR/R^{k-1})$ factor was proven by Austrin and Mossel [2] and, independently, by Guruswami and Raghavendra [16]. For the case $k = 2$, results by Khot et al. [22] implicitly demonstrate UGC-hardness of approximation within $O(\log R/R)$, made explicit in [24]. Moreover, Austrin and Mossel proved UGC-hardness of approximation of $O(k/R^{k-1})$ for infinitely many ks [2], but in the regime $k \geq R$. Recently, Chan was able to remove the Unique Game Conjecture assumption from these results [3]. More specifically, Chan showed NP-hardness of approximation of factor $O(kR/R^{k-1})$ for every k, R and that of factor $O(k/R^{k-1})$ for every $k \geq R$. Due to an approximation algorithm with matching approximation ratio by Makarychev and Makarychev [26], Chan's result established tight hardness of approximation for $k \geq R$. On the other hand, when $k < R$, Chan's result gives $O(kR/R^{k-1})$ hardness of approximation whereas the best known approximation algorithm achieves only $\Omega(k/R^{k-1})$ approximation ratio [26, 14]. In an attempt to bridge this gap, we prove the following theorem.

▶ **Theorem 1** (Main Theorem). *Assuming the Unique Games Conjecture, it is NP-hard to approximate* Max k-CSP$_R$ *to within* $2^{O(k \log k)}(\log R)^{k/2}/R^{k-1}$ *factor, for any* $k \geq 2$ *and any sufficiently large* R.

When $k = o(\log R/\log\log R)$, our result improves upon the previous best known hardness of approximation result in this regime, due to Chan. In particular, when k is constant, our results are tight up to a factor of $O(\text{polylog } R)$. While Chan's results hold unconditionally, our result, similar to many of the aforementioned results (e.g. [33, 2, 16]), rely on the Unique Games Conjecture.

A *unique game* is a Max 2-CSP instance where each constraint is a permutation. The *Unique Games Conjecture (UGC)*, first proposed by Khot in his seminal paper in 2002 [20], states that, for any sufficiently small $\eta, \gamma > 0$, it is NP-hard to distinguish a unique game where at least $1 - \eta$ fraction of constraints can be satisfied from a unique game where at most γ fraction of constraints can be satisfied. The UGC has since made a huge impact in hardness of approximation; numerous hardness of approximation results not known unconditionally can be derived assuming the UGC. More surprisingly, UGC-hardness of approximation for

Range of k, R	NP-Hardness	UGC-Hardness	Approximation	References
$k = 2$	$O\left(\frac{\log R}{\sqrt{R}}\right)$	$O\left(\frac{\log R}{R}\right)$	$\Omega\left(\frac{\log R}{R}\right)$	[3, 22, 24]
$3 \leq k < R$	$O\left(\frac{k}{R^{k-2}}\right)$	–	$\Omega\left(\frac{k}{R^{k-1}}\right)$	[3, 26, 14]
$R \leq k$	$O\left(\frac{k}{R^{k-1}}\right)$	–	$\Omega\left(\frac{k}{R^{k-1}}\right)$	[3, 26]
Any k, R	–	$\frac{2^{O(k \log k)}(\log R)^{k/2}}{R^{k-1}}$	$\Omega\left(\frac{\log R}{R^{k-1}}\right)$	this work

Figure 1 Comparison between our work and previous works. We list the previous best known results alongside our results. From previous works, there is either an NP-hardness or a UGC-hardness result matching the best known approximation algorithm in every case except when $3 \leq k < R$. Our hardness result improves on the best known hardness result when $k = o(\log R/\log\log R)$, and our approximation algorithm improves on the previously known algorithm when $k = o(\log R)$.

various problems, such as Max Cut [22], Vertex Cover [23] and *any* Max CSP [31][1], are known to be tight. For more details on UGC and its implications, we refer interested readers to Khot's survey [21] on the topic.

Another related conjecture from [20] is the *d-to-1 Conjecture*. In the d-to-1 Conjecture, we consider *d-to-1 games* instead of unique games. A d-to-1 game is an instance of Max 2-CSP where the constraint graph is bipartite. Moreover, each constraint must be a d-to-1 function, i.e., for each assignment to a variable on the right, there exists d assignments to the corresponding variable on the left that satisfy the constraint. The d-to-1 Conjecture states that, for any sufficiently small $\gamma > 0$, it is NP-hard to distinguish between a fully satisfiable d-to-1 game and a d-to-1 game where at most γ fraction of constraints can be satisfied. Currently, it is unknown whether the d-to-1 Conjecture implies the Unique Games Conjecture and vice versa.

While the d-to-1 Conjecture has yet to enjoy the same amount of influence as the UGC, it has been proven successful in providing a basis for hardness of graph coloring problems [9, 10, 17] and for Max 3-CSP with perfect completeness [30, 34]. Here we show that, by assuming the d-to-1 Conjecture instead of UGC, we can get a similar hardness of approximation result for Max k-CSP$_R$ as stated below.

▶ **Theorem 2.** *Assuming the d-to-1 Games Conjecture holds for some d, it is NP-hard to approximate* Max k-CSP$_R$ *to within $2^{O(k \log k)}(\log R)^{k/2}/R^{k-1}$ factor, for any $k \geq 2$ and any sufficiently large R.*

As mentioned earlier, there has also been a long line of works in approximation algorithms for Max k-CSP$_R$. In the boolean case, Trevisan first showed a polynomial-time approximation algorithm with approximation ratio $2/2^k$ [35]. Hast later improved the ratio to $\Omega(k/(2^k \log k))$ [18]. Charikar, Makarychev and Makarychev then came up with an $\Omega(k/2^k)$-approximation algorithm [6]. As stated when discussing hardness of approximation of Max k-CSP$_2$, this approximation ratio is tight up to a constant factor.

For the non-boolean case, Charikar, Makarychev, and Makarychev's algorithm achieve $\Omega(k \log R/R^k)$ ratio for Max k-CSP$_R$. Makarychev, and Makarychev later improved the

[1] Raghavendra showed in [31] that it is hard to approximate any Max CSP beyond what a certain type of semidefinite program can achieve. However, determining the approximation ratio of a semidefinite program is still not an easy task. Typically, one still needs to find an integrality gap for such a program in order to establish the approximation ratio.

approximation ratio to $\Omega(k/R^{k-1})$ when $k = \Omega(\log R)$ [26]. Additionally, the algorithm was extended by Goldshlager and Moshkovitz to achieve the same approximation ratio for any k, R [14]. On this front, we show the following result.

▶ **Theorem 3.** *There exists a polynomial-time* $\Omega(\log R/R^{k-1})$*-approximation algorithm for* Max k-CSP$_R$.

In comparison to the previous algorithms, our algorithm gives better approximation ratio than all the known algorithms when $k = o(\log R)$. We remark that our algorithm is just a simple extension of Kindler, Kolla and Trevisan's polynomial-time $\Omega(\log R/R)$-approximation algorithm for Max 2-CSP$_R$ [24].

1.1 Summary of Techniques

Our reduction from Unique Games to Max k-CSP$_R$ follows the reduction of [22] for Max 2-CSPs. We construct a k-query PCP using a Unique-Label-Cover "outer verifier", and then design a k-query Long Code test as an "inner verifier". For simplicity, let us think of k as a constant. We essentially construct a k-query *Dictator-vs.-Quasirandom* test for functions $f : [R]^n \to [R]$, with completeness $\Omega(1/(\log R)^{k/2})$ and soundness $O(1/R^{k-1})$. Our test is structurally similar to the 2-query "noise stability" tests of [22]: first we pick a random $z \in [R]^n$, then we pick k weakly-correlated queries $x^{(1)}, \dots, x^{(k)}$ by choosing each $x^{(i)} \in [R]^n$ as a noisy copy of z, i.e., each coordinate $(x^{(i)})_j$ is chosen as z_j with some probability ρ or uniformly at random otherwise. We accept iff $f(x^{(1)}) = f(x^{(2)}) = \cdots = f(x^{(k)})$. The key technical step is our analysis of the soundness of this test. We need to show that if f is a balanced function with small low-degree influences, then the test passes with probability $O(1/R^{k-1})$. Intuitively, we would like to say that for high enough noise, the values $f(x^{(i)})$ are roughly independent and uniform, so the test passes with probability around $1/R^{k-1}$. To formalize this intuition, we utilize the *Invariance Principle* and *Hypercontractivity.*

More precisely, if we let $f^i(x) : [R]^n \to \{0, 1\}$ be the indicator function for $f(x) = i$, then the test accepts iff $f^i(x^{(1)}) = \cdots = f^i(x^{(k)}) = 1$ for some $i \in [R]$. For each $i \in [R]$, this probability can be written as the expectation of the product: $\mathbb{E}[f^i(x^{(1)})f^i(x^{(2)}) \dots f^i(x^{(k)})]$. Since $x^{(i)}$'s are chosen as noisy copies of z, this expression is related to the k-th norm of a noisy version of f^i. Thus, our problem is reduced to bounding the k-norm of a noisy function $\widetilde{f^i} : [R]^n \to [0, 1]$, which has bounded one-norm $\mathbb{E}[\widetilde{f^i}] = 1/R$ since f is balanced. To arrive at this bound, we first apply the Invariance Principle, which essentially states that a low-degree low-influence function on $[R]^n$ behaves on random input similarly to its "boolean analog" over domain $\{\pm 1\}^{nR}$. Here "boolean analog" refers to the function over $\{\pm 1\}^{nR}$ with matching Fourier coefficients.

Roughly speaking, now that we have transfered to the boolean domain, we are left to bound the k-norm of a noisy function on $\{\pm 1\}^{nR}$ based on its one-norm. This follows from Hypercontractivity in the boolean setting, which bounds a higher norm of any noisy function on boolean domain in terms of a lower norm.

The description above hides several technical complications. For example, when we pass from a function $[R]^n \to [0, 1]$ to its "boolean analog" $\{\pm 1\}^{nR} \to \mathbb{R}$, the range of the resulting function is no longer bounded to $[0, 1]$. This, along with the necessary degree truncation, means we must be careful when bounding norms. Further, Hypercontractivity only allows us to pass from k-norms to $(1 + \varepsilon)$-norms for small ε, so we cannot use the known 1-norm directly. For details on how we handle these issues, see Section 3. This allows us to prove soundness of our dictator test without passing through results on Gaussian space, as was done to prove the "Majority is Stablest" conjecture [27] at the core of the [22] 2-query dictator test.

To extend our result to work with d-to-1 Games Conjecture in place of UGC, we observe that our proof goes through even when we assume a conjecture weaker than the UGC, which we name the *One-Sided Unique Games Conjecture*. The only difference between the original UGC and the One-Sided UGC is that the completeness in UGC is allowed to be any constant smaller than one but the completeness is a fixed constant for the One-Sided UGC. The conjecture is formalized as Conjecture 13. We show that the d-to-1 Games Conjecture also implies the One-Sided UGC, which means that our inapproximability result for Max k-CSP$_R$ also holds when we change our assumption to d-to-1 Games.

Lastly, for our approximation algorithm, we simply extend the Kindler, Kolla and Trevisan's algorithm by first creating an instance of Max 2-CSP$_R$ from Max k-CSP$_R$ by projecting each constraint to all possible subsets of two variables. We then use their algorithm to approximate the instance. Finally, we set our assignment to be the same as that from KKT algorithm with some probability. Otherwise, we pick the assignment uniformly at random from $[R]$. As we shall show in Section 4, with the right probability, this technique can extend not only the KKT algorithm but any algorithm for Max k'-CSP$_R$ to an algorithm for Max k-CSP$_R$ where $k > k'$ with some loss in the advantage over the naive randomized algorithm.

1.2 Organization of the Paper

In Section 2, we define notations and list background knowledge that will be used throughout the paper. Next, we prove our hardness of approximation results, i.e., Theorem 1 and Theorem 2, in Section 3. In Section 4, we show how to extend Kindler et al.'s algorithm to Max k-CSP$_R$ and prove Theorem 3. We also explicitly present the dictator test that is implicit in our hardness proof, in Section 5. Finally, in Section 6, we discuss interesting open questions and directions for future works.

2 Preliminaries

In this section, we list notations and previous results that will be used to prove our results.

2.1 Max k-CSP$_R$

We start by giving a formal definition of Max k-CSP$_R$, which is the main focus of our paper.

▶ **Definition 4** (Max k-CSP$_R$)**.** An instance $(\mathcal{X}, \mathcal{C})$ of (weighted) Max k-CSP$_R$ consists of

- A set $\mathcal{X} = \{x_1, \dots, x_n\}$ of variables.
- A set $\mathcal{C} = \{C_1, \dots, C_m\}$ of constraints. Each constraint C_i is a triple (W_i, S_i, P_i) of a positive weight $W_i > 0$ such that $\sum_{i=1}^m W_i = 1$, a subset of variables $S_i \subseteq \mathcal{X}$ of size k, and a predicate $P_i : [R]^{S_i} \to \{0, 1\}$ that maps each assignment to variables in S_i to $\{0, 1\}$. Here we use $[R]^{S_i}$ to denote the set of all functions from S_i to $[R]$, i.e., $[R]^{S_i} = \{\psi : S_i \to [R]\}$.

For each assignment of variables $\varphi : \mathcal{X} \to [R]$, we define its value to be the total weights of the predicates satisfied by this assignment, i.e., $\sum_{i=1}^m W_i P_i(\varphi |_{S_i})$. The goal is to find an assignment $\varphi : \mathcal{X} \to [R]$ that with maximum value. We sometimes call the optimum the value of $(\mathcal{X}, \mathcal{C})$.

Note that, while the standard definition of Max k-CSP$_R$ refers to the unweighted version where $W_1 = \cdots = W_m = 1/m$, Crescenzi, Silvestri and Trevisan showed that the approximability of these two cases are essentially the same [8].[2] Hence, it is enough for us to consider just the weight version.

2.2 Unique Games and d-to-1 Conjectures

In this subsection, we give formal definitions for unique games, d-to-1 games and Khot's conjectures about them. First, we give a formal definition of unique games.

▶ **Definition 5** (Unique Game). A unique game $(V, W, E, N, \{\pi_e\}_{e \in E})$ consists of

- A bipartite graph $G = (V, W, E)$.
- Alphabet size N.
- For each edge $e \in E$, a permutation $\pi_e : [N] \to [N]$.

For an assignment $\varphi : V \cup W \to [N]$, an edge $e = (v, w)$ is satisfied if $\pi_e(\varphi(v)) = \varphi(w)$. The goal is to find an assignment that satisfies as many edges as possible. We define the value of an instance to be the fraction of edges satisfied in the optimum solution.

The Unique Games Conjecture states that it is NP-hard to distinguish an instance of value close one from that of value almost zero. More formally, it can be stated as follows.

▶ **Conjecture 6** (Unique Games Conjecture). *For every constant $\eta, \gamma > 0$, there exists a constant $N = N(\eta, \gamma)$ such that it is NP-hard to distinguish a unique game with alphabet size N of value at least $1 - \eta$ from one of value at most γ.*

Next, we define d-to-1 games, which is similar to unique games except that each constraint is a d-to-1 function instead of a permutation.

▶ **Definition 7** (d-to-1 Game). A d-to-1 game $(V, W, E, N, \{\pi_e\}_{e \in E})$ consists of

- A bipartite graph $G = (V, W, E)$.
- Alphabet size N.
- For each edge $e \in E$, a function $\pi_e : [N] \to [N/d]$ such that $|\pi_e^{-1}(\sigma)| = d$ for every $\sigma \in [N/d]$.

Satisfiability of an edge, the goal, and an instance's value of is defined similar to that of unique games.

In contrast to the UGC, d-to-1 Conjecture requires perfect completeness, i.e., it states that we cannot distinguish even a full satisfiable d-to-1 game from one with almost zero value.

▶ **Conjecture 8** (d-to-1 Conjecture). *For every constant $\gamma > 0$, there exists a constant $N = N(\gamma)$ such that it is NP-hard to distinguish a d-to-1 game with alphabet size N of value 1 from one of value at most γ.*

[2] More specifically, they proved that, if the weighted version is NP-hard to approximate to within ratio r, then the unweighted version is also NP-hard to approximate to within $r - o_n(1)$ where $o_n(1)$ represents a function such that $o_n(1) \to 0$ as $n \to \infty$.

2.3 Fourier Expansions

For any function $g : [q]^n \to \mathbb{R}$ over a finite alphabet $[q]$, we define the Fourier expansion of g as follows.

Consider the space of all functions $[q] \to \mathbb{R}$, with the inner-product $\langle u, v\rangle := \mathbb{E}_{x\in[q]}[u(x)v(x)]$, where the expectation is over a uniform $x \in [q]$. Pick an orthonormal basis $l_1, \dots, l_q$ for this space $l_i : \Sigma \to \mathbb{R}$, such that l_1 is the constant function 1. We can now write g in the tensor-product basis, as

$$g(x_1, x_2, \dots, x_n) = \sum_{s\in[q]^n} \hat{g}(s) \cdot \prod_{i=1}^{n} l_{s(i)}(x_i). \tag{1}$$

Since we pick $l_1(x) = 1$ for all $x \in [q]$, we also have $\mathbb{E}_{x\in[q]}[l_i(x)] = \langle l_i, l_1\rangle = 0$ for every $i \neq 1$.

Throughout, we use $\hat{g}$ to refer to the Fourier coefficients of a function g.

For functions $g : [q]^n \to \mathbb{R}$, the p-norm is defined as

$$||g||_p = \mathop{\mathbb{E}}_{x\in[q]^n}[|g(x)|^p]^{1/p}. \tag{2}$$

In particular, the squared 2-norm is

$$||g||_2^2 = \mathop{\mathbb{E}}_{x\in[q]^n}[g(x)^2] = \sum_{s\in[q]^n} \hat{g}(s)^2. \tag{3}$$

2.3.1 Noise Operators

For $x \in [q]^n$, let $y \overset{\rho}{\leftarrow} x$ denote that y is a ρ-correlated copy of x. That is, each coordinate y_i is independently chosen to be equal to x_i with probability ρ, or chosen uniformly at random otherwise.

Define the noise operator T_ρ acting on any function $g : [q]^n \to \mathbb{R}$ as

$$(T_\rho g)(x) = \mathop{\mathbb{E}}_{y\overset{\rho}{\leftarrow}x}[g(y)]. \tag{4}$$

Notice that the noise operator T_ρ acts on the Fourier coefficients on this basis as follows.

$$f(x) = T_\rho g(x) = \sum_{s\in[q]^n} \hat{g}(s) \cdot \rho^{|s|} \cdot \prod_{i=1}^{n} l_{s(i)}(x_i) \tag{5}$$

where $|s|$ is defined as $|\{i \mid s(i) \neq 1\}|$.

2.3.2 Degree Truncation

For any function $g : [q]^n \to \mathbb{R}$, let $g^{\leq d}$ denote the $(\leq d)$-degree part of g, i.e.,

$$g^{\leq d}(x) = \sum_{s\in[q]^n, |s|\leq d} \hat{g}(s) \cdot \prod_{i=1}^{n} l_{s(i)}(x_i), \tag{6}$$

and similarly let $g^{>d} : [q]^n \to \mathbb{R}$ denote the $(> k)$-degree part of g, i.e.,

$$g^{>d}(x) = \sum_{s\in[q]^n, |s|>d} \hat{g}(s) \cdot \prod_{i=1}^{n} l_{s(i)}(x_i). \tag{7}$$

Notice that degree-truncation commutes with the noise-operator, so writing $T_\rho g^{\leq d}$ is unambiguous.

Also, notice that truncating does not increase 2-norm:

$$||g^{\leq d}||_2^2 = \sum_{s \in [q]^n, |s| \leq d} \hat{g}(s)^2 \leq \sum_{s \in [q]^n} \hat{g}(s)^2 = ||g||_2^2. \tag{8}$$

We frequently use the fact that noisy functions have small high-degree contributions. That is, for any function $g : [q]^n \to [0,1]$, we have

$$||T_\rho g^{>d}||_2^2 = \sum_{s \in [q]^n, |s| > d} \rho^{2|s|} \hat{g}(s)^2 \leq \rho^{2d} \sum_{s \in [q]^n} \hat{g}(s)^2 = \rho^{2d} ||g||_2^2 \leq \rho^{2d}. \tag{9}$$

2.3.3 Influences

For any function $g : [q]^n \to \mathbb{R}$, the influence of coordinate $i \in [n]$ is defined as

$$Inf_i[g] = \mathop{\mathbb{E}}_{x \in [q]^n} [Var_{x_i \in [q]}[g(x) \mid \{x_j\}_{j \neq i}]]. \tag{10}$$

This can be expressed in terms of the Fourier coefficients of g as follows:

$$Inf_i[g] = \sum_{s \in [q]^n:\ s(i) \neq 1} \hat{g}(s)^2. \tag{11}$$

Further, the degree-d influences are defined as

$$Inf_i^{\leq d}[g] = Inf_i[g^{\leq d}] = \sum_{\substack{s \in [q]^n: \\ |s| \leq d,\ s(i) \neq 1}} \hat{g}(s)^2. \tag{12}$$

2.3.4 Binary Functions

The previous discussion of Fourier analysis can be applied to boolean functions, by specializing to $q = 2$. However, in this case the Fourier expansion can be written in a more convenient form. Let $G : \{+1, -1\}^n \to \mathbb{R}$ be a boolean function over n bits. We can choose orthonormal basis functions $l_1(y) = 1$ and $l_2(y) = y$, so G can be written as

$$G(y) = \sum_{S \subseteq [n]} \hat{G}(S) \prod_{i \in S} y_i \tag{13}$$

for some coefficients $\hat{G}(S)$.

Degree-truncation then results in

$$G^{\leq d}(y) = \sum_{S \subseteq [n] : |S| \leq d} \hat{G}(S) \prod_{i \in S} y_i, \tag{14}$$

and the noise-operator acts as follows:

$$T_\rho G(y) = \sum_{S \subseteq [n]} \hat{G}(S) \rho^{|S|} \prod_{i \in S} y_i. \tag{15}$$

2.3.5 Boolean Analogs

To analyze k-CSP$_R$, we will want to translate between functions over $[R]^n$ to functions over $\{\pm 1\}^{nR}$. The following notion of *boolean analogs* will be useful.

For any function $g : [R]^n \to \mathbb{R}$ with Fourier coefficients $\hat{g}(s)$, define the boolean analog of g to be a function $\{g\} : \{\pm 1\}^{n \times R} \to \mathbb{R}$ such that

$$\{g\}(z) = \sum_{s \in [R]^n} \hat{g}(s) \cdot \prod_{i \in [n], s(i) \neq 1} z_{i,s(i)}. \tag{16}$$

Note that

$$||g||_2^2 = \sum_{s \in [R]^n} \hat{g}(s)^2 = ||\{g\}||_2^2, \tag{17}$$

and that

$$\{g^{\leq d}\} = \{g\}^{\leq d}. \tag{18}$$

Moreover, the noise operator acts nicely on $\{g\}$ as follows:

$$T_\rho \{g\} = \{T_\rho g\}. \tag{19}$$

For simplicity, we use T_ρ to refer to both boolean and non-boolean noise operators with correlation ρ.

2.4 Invariance Principle and Mollification Lemma

We start with the Invariance Principle in the form of Theorem 3.18 in [27]:

▶ **Theorem 9** (General Invariance Principle [27]). *Let $f : [R]^n \to \mathbb{R}$ be any function such that $Var[f] \leq 1$, $Inf_i[f] \leq \delta$, and $deg(f) \leq d$. Let $F : \{\pm 1\}^{nR} \to \mathbb{R}$ be its boolean analog: $F = \{f\}$. Consider any "test-function" $\psi : \mathbb{R} \to \mathbb{R}$ that is $\mathcal{C}^3$, with bounded 3rd-derivative $|\psi'''| \leq C$ everywhere. Then,*

$$\left| \mathop{\mathbb{E}}_{y \in \{\pm 1\}^{nR}} [\psi(F(y))] - \mathop{\mathbb{E}}_{x \in [R]^n} [\psi(f(x))] \right| \leq C 10^d R^{d/2} \sqrt{\delta}. \tag{20}$$

Note that the above version follows directly from Theorem 3.18 and Hypothesis 3 of [27], and the fact that uniform ± 1 bits are $(2, 3, 1/\sqrt{2})$-hypercontractive as described in [27].

As we shall see later, we will want to apply the Invariance Principle for some functions ψ that are not in $\mathcal{C}^3$. However, these functions will be Lipschitz-continuous with some constant $c \in \mathbb{R}$ (or "c-Lipschitz"), meaning that

$$|\psi(x + \Delta) - \psi(x)| \leq c|\Delta| \quad \text{for all } x, \Delta \in \mathbb{R}. \tag{21}$$

Therefore, similar to Lemma 3.21 in [27], we can "smooth" it to get a function $\widetilde{\psi}$ that is that is $\mathcal{C}^3$, and has arbitrarily small pointwise difference to ψ.

▶ **Lemma 10** (Mollification Lemma [27]). *Let $\psi : \mathbb{R} \to \mathbb{R}$ be any c-Lipschitz function. Then for any $\zeta > 0$, there exists a function $\widetilde{\psi} : \mathbb{R} \to \mathbb{R}$ such that*
- $\widetilde{\psi} \in \mathcal{C}^3$,
- $||\widetilde{\psi} - \psi||_\infty \leq \zeta$, *and,*
- $||\widetilde{\psi}'''||_\infty \leq \widetilde{C} c^3 / \zeta^2$.

For some universal constant $\widetilde{C}$, not depending on ζ, c.

For completeness, the full proof of the lemma can be found in Appendix A.1.

Now we state the following version of the Invariance Principle, which will be more convenient to invoke. It can be proved simply by just combining the two previous lemmas. We list a full proof in Appendix A.2.

▶ **Lemma 11** (Invariance Principle). *Let $\psi : \mathbb{R} \to \mathbb{R}$ be one of the following functions:*

1. $\psi_1(t) := |t|$,
2. $\psi_k(t) := \begin{cases} t^k & \text{if } t \in [0,1], \\ 0 & \text{if } t < 0, \\ 1 & \text{if } t \geq 1. \end{cases}$

Let $f : [R]^n \to [0,1]$ be any function with all $Inf_i^{\leq d}[f] \leq \delta$. Let $F : \{\pm 1\}^{nR} \to \mathbb{R}$ be its boolean analog: $F = \{f\}$. Let $f^{\leq d} : [R]^n \to \mathbb{R}$ denote f truncated to degree d, and similarly for $F^{\leq d} : \{\pm 1\}^{nR} \to \mathbb{R}$.

Then, for parameters $d = 10k \log R$ and $\delta = 1/(R^{10+100k\log(R)})$, we have

$$\left| \mathop{\mathbb{E}}_{y \in \{\pm 1\}^{nR}} [\psi(F^{\leq d}(y))] - \mathop{\mathbb{E}}_{x \in [R]^n} [\psi(f^{\leq d}(x))] \right| \leq O(1/R^k). \tag{22}$$

2.5 Hypercontractivity Theorem

Another crucial ingredient in our proof is the hypercontractivity lemma, which says that, on $\{\pm 1\}^n$ domain, the operator T_ρ smooths any function so well that the higher norm can be bound by the lower norm of the original (unsmoothed) function. Here we use the version of the theorem as stated in [28].

▶ **Theorem 12** (Hypercontractivity Theorem [28]). *Let $1 \leq p \leq q \leq \infty$. For any $\rho \leq \sqrt{\frac{p-1}{q-1}}$ and for any function $h : \{\pm 1\}^n \to \mathbb{R}$, the following inequality holds:*

$$||T_\rho h||_q \leq ||h||_p. \tag{23}$$

In particular, for choice of parameter $\rho = 1/\sqrt{(k-1)\log R}$, we have

$$||T_{2\rho} h||_k \leq ||h||_{1+\varepsilon}. \tag{24}$$

where $\varepsilon = 4/\log(R)$.

3 Inapproximability of Max k-CSP$_R$

In this section, we prove Theorem 1 and Theorem 2. Before we do so, we first introduce a conjecture, which we name *One-Sided Unique Games Conjecture*. The conjecture is similar to UGC except that the completeness parameter ζ is fixed in contrast to UGC where the completeness can be any close to one.

▶ **Conjecture 13** (One-Sided Unique Games Conjecture). *There exists a constant $\zeta > 0$ such that, for every constant $\gamma > 0$, there exists a constant $N = N(\gamma)$ such that it is NP-hard to distinguish a unique game with alphabet size N of value at least ζ from one of value at most γ.*

It is obvious that the UGC implies One-Sided UGC with $\zeta = 1 - \eta$ for any sufficiently small η. It is also not hard to see that, by repeating each alphabet on the right d times and spreading each d-to-1 constraint out to be a permutation, d-to-1 Games Conjecture implies One-Sided UGC with $\zeta = 1/d$. A full proof of this can be found in Appendix B.

Since both UGC and d-to-1 Games Conjecture imply One-Sided UGC, it is enough for us to show the following theorem, which implies both Theorem 1 and Theorem 2.

▶ **Theorem 14.** *Unless the One-Sided Unique Games Conjecture is false, for any $k \geq 2$ and any sufficiently large R, it is NP-hard to approximate* Max k-CSP$_R$ *to within $2^{O(k \log k)}(\log R)^{k/2}/R^{k-1}$ factor.*

The theorem will be proved in Subsection 3.3. Before that, we first prove an inequality that is the heart of our soundness analysis in Subsection 3.2.

3.1 Parameters

We use the following parameters throughout, which we list for convenience here:

- Correlation[3]: $\rho = 1/\sqrt{(k-1)\log R}$
- Degree: $d = 10k \log R$
- Low-degree influences: $\delta = 1/(R^{10+100k\log(R)})$

3.2 Hypercontractivity for Noisy Low-Influence Functions

Here we show a version of hypercontractivity for noisy low-influence functions over large domains. Although hypercontractivity does not hold in general for noisy functions over large domains, it turns out to hold in our setting of high-noise and low-influences. The main technical idea is to use the Invariance Principle to reduce functions over larger domains to boolean functions, then apply boolean hypercontractivity.

▶ **Lemma 15** (Main Lemma). *Let $g : [R]^n \to [0,1]$ be any function with $\mathbb{E}_{x\in[R]^n}[g(x)] = 1/R$. Then, for our choice of parameters ρ, d, δ: If $Inf_i^{\leq d}[g] \leq \delta$ for all i, then*

$$\mathop{\mathbb{E}}_{x\in[R]^n}[(T_\rho g(x))^k] \leq 2^{O(k)}/R^k.$$

Before we present the full proof, we outline the high-level steps below. Let $f = T_\rho g$, and define boolean analogs $G = \{g\}$, and $F = \{f\}$. Let $\psi_k : \mathbb{R} \to \mathbb{R}$ be defined as in Lemma 11. Then,

$$\begin{aligned}
&& \mathop{\mathbb{E}}_{x\in[R]^n}[f(x)^k] &\approx \mathbb{E}[\psi_k(f^{\leq d}(x))] && (25)\\
&\text{(Lemma 11: Invariance Principle)} && \approx \mathop{\mathbb{E}}_{y\in\{\pm1\}^{nR}}[\psi_k(F^{\leq d}(y))] && (26)\\
&\text{(Definition of } \psi_k) && \leq ||F^{\leq d}||_k^k && (27)\\
&\text{(Definition of } F) && = ||T_\rho G^{\leq d}||_k^k && (28)\\
&&& = ||T_{2\rho}T_{1/2}G^{\leq d}||_k^k && (29)\\
&\text{(Hypercontractivity, for small } \varepsilon) && \leq ||T_{1/2}G^{\leq d}||_{1+\varepsilon}^k && (30)\\
&\text{(Invariance, etc.)} && \approx 2^{O(k)}||g||_1^k && (31)\\
&\text{(Since } \mathbb{E}[|g|] = 1/R) && = 2^{O(k)}/R^k. && (32)
\end{aligned}$$

[3] Note that for $k = 2$, this correlation yields a stability of $\approx 1/R$ for the plurality. That is, $\Pr_{z,x,y}[\text{plur}(x_1,\ldots,x_n) = \text{plur}(y_1,\ldots,y_n)] \approx 1/R$ where each $z_i \in [R]$ is iid uniform, and x_i, y_i are ρ-correlated copies of z_i.

Proof. To establish line (25), first notice that

$$\psi_k(f(x)) = \psi_k(f^{\leq d}(x) + f^{>d}(x)) \leq \psi_k(f^{\leq d}(x)) + k|f^{>d}(x)| \tag{33}$$

where the last inequality is because the function ψ_k is k-Lipschitz.

Moreover, since $g(x) \in [0,1]$, we have $f(x) \in [0,1]$, so

$$f(x)^k = \psi_k(f(x)). \tag{34}$$

Thus,

$$\begin{aligned}
\mathbb{E}[f(x)^k] &= \mathbb{E}[\psi_k(f(x))] && (35)\\
&\leq \mathbb{E}[\psi_k(f^{\leq d}(x))] + k\,\mathbb{E}[|f^{>d}(x)|] && (36)\\
&= \mathbb{E}[\psi_k(f^{\leq d}(x))] + k||f^{>d}||_1 && (37)\\
&\leq \mathbb{E}[\psi_k(f^{\leq d}(x))] + k||f^{>d}||_2. && (38)\\
& && (39)
\end{aligned}$$

And we can bound the 2-norm of $f^{>d}$, since f is noisy, we have

$$||f^{>d}||_2^2 = ||T_\rho g^{>d}||_2^2 \leq \rho^{2d} \leq O(1/R^{2k}). \tag{40}$$

The last inequality comes from our choice of ρ and d.

So line (25) is established:

$$\mathbb{E}[f(x)^k] \leq \mathbb{E}[\psi_k(f^{\leq d}(x))] + O(k/R^k). \tag{41}$$

Line (26) follows directly from our version of the Invariance Principle (Lemma 11), for the function ψ_k:

$$\mathop{\mathbb{E}}_{x\in[R]^n}[\psi_k(f^{\leq d}(x))] \leq \mathop{\mathbb{E}}_{y\in\{\pm1\}^{nR}}[\psi_k(F^{\leq d}(y))] + O(1/R^k). \tag{42}$$

We can now rewrite $\mathbb{E}_{y\in\{\pm1\}^{nR}}[\psi_k(F^{\leq d}(y))]$ as

$$\begin{aligned}
\mathop{\mathbb{E}}_{y\in\{\pm1\}^{nR}}[\psi_k(F^{\leq d}(y))] &\leq \mathop{\mathbb{E}}_{y\in\{\pm1\}^{nR}}[|F^{\leq d}(y)|^k] && (43)\\
&= ||F^{\leq d}||_k^k && (44)\\
&= ||T_\rho G^{\leq d}||_k^k && (45)\\
&= ||T_{2\rho}T_{1/2}G^{\leq d}||_k^k. && (46)\\
& && (47)
\end{aligned}$$

Now, from the Hypercontractivity Theorem, Equation (24), we have

$$||T_{2\rho}T_{1/2}G^{\leq d}||_k \leq ||T_{1/2}G^{\leq d}||_{1+\varepsilon} \tag{48}$$

for $\varepsilon = 4/\log R$. This establishes line (30):

$$||T_{2\rho}T_{1/2}G^{\leq d}||_k^k \leq ||T_{1/2}G^{\leq d}||_{1+\varepsilon}^k = \mathbb{E}[|T_{1/2}G^{\leq d}(y)|^{1+\varepsilon}]^{k/(1+\varepsilon)}. \tag{49}$$

To show the remaining steps, we will apply the Invariance Principle once more. Notice that for all $t \in \mathbb{R} : |t|^{1+\varepsilon} \leq |t| + t^2$. Hence, we can derive the following bound:

$$\begin{aligned}
& \mathbb{E}[|T_{1/2}G^{\leq d}(y)|^{1+\varepsilon}] \leq \mathbb{E}[|T_{1/2}G^{\leq d}(y)|] + \mathbb{E}[(T_{1/2}G^{\leq d}(y))^2] && (50)\\
\text{(Matching Fourier expansion)}\quad & = \mathbb{E}[|T_{1/2}G^{\leq d}(y)|] + \mathbb{E}[(T_{1/2}g^{\leq d}(y))^2] && (51)\\
\text{(Lemma 11, Invariance Principle)}\quad & \leq \mathbb{E}[|T_{1/2}g^{\leq d}(x)|] + \mathbb{E}[(T_{1/2}g^{\leq d}(x))^2] + O(1/R^k). && (52)
\end{aligned}$$

Here we applied our Invariance Principle (Lemma 11) for the function ψ_1 as defined in Lemma 11. We will bound each of the expectations on the RHS, using the fact that g is balanced, and $T_{1/2}g$ is noisy.

First,

$$\begin{aligned}
& \mathbb{E}[|T_{1/2}g^{\le d}(x)|] = \mathbb{E}[|T_{1/2}g(x) - T_{1/2}g^{>d}(x)|] && (53)\\
\text{(Triangle Inequality)} \quad & \le \mathbb{E}[|T_{1/2}g(x)|] + \mathbb{E}[|T_{1/2}g^{>d}(x)|] && (54)\\
& = ||g||_1 + ||T_{1/2}g^{>d}||_1 && (55)\\
& \le ||g||_1 + ||T_{1/2}g^{>d}||_2 && (56)\\
& \le 1/R + (1/2)^d && (57)\\
\text{(By our choice of } d) \quad & = O(1/R). && (58)
\end{aligned}$$

Second,

$$\begin{aligned}
& \mathbb{E}[(T_{1/2}g^{\le d}(x))^2] = \sum_{s\in[R]^n, |s|\le d} (1/2)^{2|s|}\hat{g}(s)^2 && (59)\\
& \le \sum_{s\in[R]^n} (1/2)^{2|s|}\hat{g}(s)^2 && (60)\\
& = \mathbb{E}[(T_{1/2}g(x))^2] && (61)\\
\text{(Since } g \in [0,1]) \quad & \le \mathbb{E}[T_{1/2}g(x)] && (62)\\
& = \mathbb{E}[g(x)] = 1/R. && (63)
\end{aligned}$$

Finally, plugging these bounds into (52), we find:

$$\begin{aligned}
& ||T_{1/2}G^{\le d}||_{1+\varepsilon}^k = \mathbb{E}[|T_{1/2}G^{\le d}(y)|^{1+\varepsilon}]^{k/(1+\varepsilon)} && (64)\\
& \le (O(1/R))^{k/(1+\varepsilon)} && (65)\\
& = 2^{O(k)}/R^{k/(1+\varepsilon)} && (66)\\
& \le 2^{O(k)}/R^{k(1-\varepsilon)} && (67)\\
\text{(Recall } \varepsilon = 4/\log R) \quad & = 2^{O(k)}/R^k. && (68)
\end{aligned}$$

This completes the proof of the main lemma. ◀

3.3 Reducing Unique Label Cover to Max k-CSP$_R$

Here we reduce unique games to Max k-CSP$_R$. We will construct a PCP verifier that reads k symbols of the proof (with an alphabet of size R) with the following properties:

- **Completeness.** If the unique game has value at least ζ, then the verifier accepts an honest proof with probability at least $c = \zeta^k/((\log R)^{k/2}2^{O(k \log k)})$.
- **Soundness.** If the unique game has value at most $\gamma = 2^{O(k)}\delta^2/(4dR^k)$, then the verifier accepts any (potentially cheating) proof with probability at most $s = 2^{O(k)}/R^{k-1}$.

Since each symbol in the proof can be viewed as a variable and each accepting predicate of the verifier can be viewed as a constraint of Max k-CSP$_R$, assuming the One-sided UGC, this PCP implies NP-hardness of approximating Max k-CSP$_R$ of factor $s/c = 2^{O(k \log k)}(\log R)^{k/2}/R^{k-1}$ and, hence, establishes our Theorem 14.

3.3.1 The PCP

Given a unique game $(V, W, E, n, \{\pi_e\}_{e \in E})$, the proof is the truth-table of a function $h_w : [R]^n \to [R]$ for each vertex $w \in W$. By folding, we can assume h_w is balanced, i.e. h_w takes on all elements of its range with equal probability: $\Pr_{x \in [R]^n}[h_w(x) = i] = 1/R$ for all $i \in [R]$.[4]

Notationally, for $x \in [R]^n$, let $(x \circ \pi)$ denote permuting the coordinates of x as: $(x \circ \pi)_i = x_{\pi(i)}$. Also, for an edge $e = (v, w)$, we write $\pi_e = \pi_{v,w}$, and define $\pi_{w,v} = \pi_{v,w}^{-1}$.

The verifier picks a uniformly random vertex $v \in V$, and k independent uniformly random neighbors of v: $w_1, w_2, \ldots, w_k \in W$. Then pick $z \in [R]^n$ uniformly at random, and let $x^{(1)}, x^{(2)}, \ldots, x^{(k)}$ be independent ρ-correlated noisy copies of z (each coordinate x_i chosen as equal to z_i w.p. ρ, or uniformly at random otherwise). The verifier accepts if and only if

$$h_{w_1}(x^{(1)} \circ \pi_{w_1,v}) = h_{w_2}(x^{(2)} \circ \pi_{w_2,v}) = \cdots = h_{w_k}(x^{(k)} \circ \pi_{w_k,v}). \tag{69}$$

To achieve the desired hardness result, we pick $\rho = 1/\sqrt{(k-1)\log R}$.

3.3.2 Completeness Analysis

First, note that that we can assume without loss of generality that the graph is regular on V side.[5] Let the degree of each vertex in V be Δ.

Suppose that the original unique game has an assignment of value at least ζ. Let us call this assignment φ. The honest proof defines h_w at each vertex $w \in W$ as the long code encoding of this assignment, i.e., $h_w(x) = x_{\varphi(w)}$. We can written the verifier acceptance condition as follows:

$$\begin{aligned}
\text{The verifier accepts} &\Leftrightarrow h_{w_1}(x^{(1)} \circ \pi_{w_1,v}) = \cdots = h_{w_k}(x^{(k)} \circ \pi_{w_k,v}) && (70)\\
&\Leftrightarrow (x^{(1)} \circ \pi_{w_1,v})_{\varphi(w_1)} = \cdots = (x^{(k)} \circ \pi_{w_k,v})_{\varphi(w_k)} && (71)\\
&\Leftrightarrow (x^{(1)})_{\pi_{w_1,v}(\varphi(w_1))} = \cdots = (x^{(k)})_{\pi_{w_k,v}(\varphi(w_k))}. && (72)
\end{aligned}$$

Observe that, if the edges $(v, w_1), \ldots, (v, w_k)$ are satisfied by φ, then $\pi_{w_1,v}(\varphi(w_1)) = \cdots = \pi_{w_k,v}(\varphi(w_k)) = \varphi(v)$. Hence, if the aforementioned edges are satisfied and $x^{(1)}, \ldots, x^{(k)}$ are not perturbed at coordinate $\varphi(v)$, then $(x^{(1)})_{\pi_{w_1,v}(\varphi(w_1))} = \cdots = (x^{(k)})_{\pi_{w_k,v}(\varphi(w_k))}$.

For each $u \in V$, let s_u be the number of satisfied edges touching u. Since $w_1, \ldots, w_k$ are chosen from the neighbors of v independently from each other, the probability that the edges $(v, w_1), (v, w_2), \ldots, (v, w_k)$ are satisfied can be bounded as follows:

$$\begin{aligned}
&\Pr_{v,w_1,\ldots,w_k}[(v, w_1), \ldots, (v, w_k) \text{ are satisfied}] && (73)\\
&= \sum_{u \in V} \Pr_{w_1,\ldots,w_k}[(v, w_1), \ldots, (v, w_k) \text{ are satisfied} \mid v = u] \Pr[v = u] && (74)\\
&= \sum_{u \in V} (s_u/\Delta)^k \Pr[v = u] && (75)\\
&= \mathop{\mathbb{E}}_{u \in V}\left[(s_u/\Delta)^k\right] && (76)\\
&\geq \mathop{\mathbb{E}}_{u \in V}[s_u/\Delta]^k. && (77)
\end{aligned}$$

[4] More precisely, if the truth-table provided is of some function $\widetilde{h}_w : [R]^n \to [R]$, we define the "folded" function h_w as $h_w(x_1, x_2, x_3, \ldots x_n) := \widetilde{h}_w(x - (x_1, x_1, \ldots, x_1)) + x_1$, where the $\pm$ is over mod R. Notice that the folded h_w is balanced, and also that folding does not affect dictator functions. Thus we define our PCP in terms of h_w, but simulate queries to h_w using the actual proof $\widetilde{h}_w$.

[5] See, for instance, Lemma 3.4 in [23].

Notice that $\mathbb{E}_{u \in V}\,[s_u/\Delta]$ is exactly the value of φ, which is at least ζ. As a result,

$$\Pr_{v,w_1,\dots,w_k}[(v,w_1),\dots,(v,w_k) \text{ are satisfied}] \geq \zeta^k.$$

Furthermore, it is obvious that the probability that $x_1,\dots,x_k$ are not perturbed at the coordinate $\varphi(v)$ is ρ^k. As a result, the PCP accepts with probability at least $\zeta^k\rho^k$. When $\rho = 1/\sqrt{(k-1)\log R}$ and ζ is a constant not depending on k and R, the completeness is $1/((\log R)^{k/2}2^{O(k\log k)})$.

3.3.3 Soundness Analysis

Suppose that the unique game has value at most $\gamma = 2^{O(k)}\delta^2/(4dR^k)$. We will show that the soundness is $2^{O(k)}/R^{k-1}$.

Suppose for the sake of contradiction that the probability that the verifier accepts $\Pr[accept] > t = 2^{\Omega(k)}/R^{k-1}$ where $\Omega(\cdot)$ hides some large enough constant.

Let $h_w^i(x) : [R]^n \to \{0,1\}$ be the indicator function for $h_w(x) = i$ and let $x \overset{\rho}{\leftarrow} z$ denote that x is a ρ-correlated copy of z. We have

$$\begin{aligned}
\Pr[accept] &= \Pr[h_{w_1}(x^{(1)} \circ \pi_{w_1,v}) = \dots = h_{w_k}(x^{(k)} \circ \pi_{w_k,v})] && (78)\\
&= \sum_{i\in[R]} \Pr[i = h_{w_1}(x^{(1)} \circ \pi_{w_1,v}) = \dots = h_{w_k}(x^{(k)} \circ \pi_{w_k,v})] && (79)\\
&= \sum_{i\in[R]} \mathbb{E}[h^i_{w_1}(x^{(1)} \circ \pi_{w_1,v}) \cdots h^i_{w_k}(x^{(k)} \circ \pi_{w_k,v})] && (80)\\
&= \sum_{i\in[R]} \mathbb{E}\left[\mathbb{E}_{w_1}[h^i_{w_1}(x^{(1)} \circ \pi_{w_1,v})] \cdots \mathbb{E}_{w_k}[h^i_{w_k}(x^{(k)} \circ \pi_{w_k,v})]\right]. && (81)
\end{aligned}$$

Where the last equality follows since the w_i's are independent, given v.

Now define $g_v^i : [R]^n \to [0,1]$ as

$$g_v^i(x) = \mathbb{E}_{w\sim v}[h_w^i(x \circ \pi_{w,v})] \tag{82}$$

where $w \sim v$ denotes neighbors w of v.

We can rewrite $\Pr[accept]$ as follows:

$$\begin{aligned}
\Pr[accept] &= \sum_{i\in[R]} \mathbb{E}[g_v^i(x^{(1)})g_v^i(x^{(2)}) \cdots g_v^i(x^{(k)})] && (83)\\
\text{(Since } x^{(j)}\text{'s are independent given } z) &= \sum_{i\in[R]} \mathbb{E}\left[\mathbb{E}_{x \overset{\rho}{\leftarrow} z}[g_v^i(x)]^k\right] && (84)\\
&= \sum_{i\in[R]} \mathbb{E}_{v,z}[(T_\rho g_v^i(z))^k] && (85)\\
&= \mathbb{E}_v\left[\sum_{i\in[R]} \mathbb{E}_z[(T_\rho g_v^i(z))^k\right]. && (86)
\end{aligned}$$

Next, notice that $\sum_{i\in[R]} \mathbb{E}_z[(T_\rho g_v^i(z))^k]$ is simply the probability the verifier accepts given it picks vertex v, and thus this sum is bounded above by 1.

Therefore, since $\Pr[accept] > t$, by (86), at least $t/2$ fraction of vertices $v \in V$ have

$$\sum_{i\in[R]} \mathbb{E}_z[(T_\rho g_v^i(z))^k] \geq t/2. \tag{87}$$

For these "good" vertices, there must exist some $i \in [R]$ for which

$$\mathop{\mathbb{E}}_z[(T_\rho g_v^i(z))^k] \geq t/(2R). \tag{88}$$

Then for "good" v and i as above,

$$\mathop{\mathbb{E}}_z[(T_\rho g_v^i(z))^k] > 2^{\Omega(k)}/R^k. \tag{89}$$

By Lemma 15 (Main Lemma), this means g_v^i has some coordinate j for which

$$Inf_j^{\leq d}[g_v^i] > \delta \tag{90}$$

for our choice of d, δ as defined in Subsection 3.1. Pick this j as the label of vertex $v \in V$.

Now to pick the label of a vertex $w \in W$, define the candidate labels as

$$Cand[w] = \{j \in [n] : \exists\ i \in [R] \text{ s.t. } Inf_j^{\leq d}[h_w^i] \geq \delta/2\}. \tag{91}$$

Notice that

$$\sum_{j\in[n]} Inf_j^{\leq d}[h_w^i] = \sum_{s\in[R]^n:\ |s|\leq d} |s|\hat{h}_w^i(s)^2 \leq d \sum_{s:|s|>0} \hat{h}_w^i(s)^2 = d\, Var[h_w^i] \leq d. \tag{92}$$

So for each $i \in [R]$, the projection h_w^i can have at most $2d/\delta$ coordinates with influence $\geq \delta/2$. Therefore the number of candidate labels is bounded:

$$|Cand[w]| \leq 2dR/\delta. \tag{93}$$

Now we argue that picking a random label in $Cand[w]$ for $w \in W$ is in expectation a good decoding. We will show that if we assigned label j to a "good" $v \in V$, then $\pi_{v,w}(j) \in Cand[w]$ for a constant fraction of neighbors $w \sim v$. Note here that $\pi_{v,w} = \pi_{w,v}^{-1}$.

First, since $g_v^i(x) = \mathbb{E}_{w\sim v}[h_w^i(x \circ \pi_{w,v})]$, the Fourier transform of g_v^i is related to the Fourier transform of the long code labels h_w^i as

$$\hat{g}_v^i(s) = \mathop{\mathbb{E}}_{w\sim v}[\hat{h}_w^i(s \circ \pi_{w,v})]. \tag{94}$$

Hence, the influence $Inf_j^{\leq d}[g_v^i]$ of being large implies the expected influence $Inf_{\pi_{v,w}^{-1}(j)}^{\leq d}[h_w^i]$ of its neighbor labels $w \sim v$ is also large as formalized below.

$$\begin{aligned}
\delta < Inf_j^{\leq k}[g_v^i] &= \sum_{\substack{s\in[R]^n\\ |s|\leq k, s_j\neq 1}} \hat{g}_v^i(s)^2 && (95)\\
&= \sum \mathop{\mathbb{E}}_{w\sim v}[\hat{h}_w^i(s \circ \pi_{w,v})]^2 && (96)\\
&\leq \sum \mathop{\mathbb{E}}_{w\sim v}[\hat{h}_w^i(s \circ \pi_{w,v})^2] && (97)\\
&= \mathop{\mathbb{E}}_{w\sim v}[\sum_{\substack{s\in[R]^n\\ |s|\leq k, s_j\neq 1}} \hat{h}_w^i(s \circ \pi_{w,v})^2] && (98)\\
&= \mathop{\mathbb{E}}_{w\sim v}[\sum_{\substack{s\in[R]^n\\ |s|\leq k, s_{\pi_{w,v}^{-1}(j)}\neq 1}} \hat{h}_w^i(s)^2] && (99)\\
\text{(Since } \pi_{v,w} = \pi_{w,v}^{-1}\text{)} &= \mathop{\mathbb{E}}_{w\sim v}[\sum_{\substack{s\in[R]^n\\ |s|\leq k, s_{\pi_{v,w}(j)}\neq 1}} \hat{h}_w^i(s)^2] && (100)\\
&= \mathop{\mathbb{E}}_{w\sim v}[Inf_{\pi_{v,w}(j)}^{\leq d}[h_w^i]] && (101)
\end{aligned}$$

Therefore, at least $\delta/2$ fraction of neighbors $w \sim v$ must have $Inf^{\leq d}_{\pi_{v,w}(j)}[h^i_w] \geq \delta/2$, and so $\pi_{v,w}(j) \in Cand[w]$ for at least $\delta/2$ fraction of neighbors of "good" vertices v.

Finally, recall that at least $(t/2)$ fraction of vertices $v \in V$ are "good". These vertices have at least $(\delta/2)$ fraction of neighbors $w \in W$ with high-influence labels and the matching label $w \in W$ is picked with probability at least $\delta/(2dR)$. Moreover, as stated earlier, we can assume that the graph is regular on V side. Hence, the expected fraction of edges satisfied by this decoding is at least

$$(t/2)(\delta/2)(\delta/2dR) = t\delta^2/(4dR) = 2^{\Omega(k)}\delta^2/(4dR^k) > \gamma, \tag{102}$$

which contradicts our assumption that the unique game has value at most γ. Hence, we can conclude that the soundness is at most $2^{O(k)}/R^{k-1}$ as desired.

4 $\Omega(\log R/R^{k-1})$-Approximation Algorithm for Max k-CSP$_R$

Instead of just extending the KKT algorithm to work with Max k-CSPs, we will show a more general statement that *any* algorithm that approximates Max CSPs with small arity can be extended to approximate Max CSPs with larger arities. In particular, we show how to extend any $f(R)/R^{k'}$-approximation algorithm for Max k'-CSP$_R$ to an $(f(R)/2^{O(\min\{k',k-k'\})})/R^k$-approximation algorithm for Max k-CSP$_R$ where $k > k'$.

Since the naive algorithm that assigns every variable randomly has an approximation ratio of $1/R^k$, we think of $f(R)$ as the advantage of algorithm A over the randomized algorithm. From this perspective, our extension lemma preserves the advantage up to a factor of $1/2^{O(\min\{k',k-k'\})}$.

The extension lemma and its proof are stated formally below.

▶ **Lemma 16.** *Suppose that there exists a polynomial-time approximation algorithm A for* Max *k'-CSP$_R$ that outputs an assignment with expected value at least $f(R)/R^{k'}$ times the optimum. For any $k > k'$, we can construct a polynomial-time approximation algorithm B for* Max *k-CSP$_R$ that outputs an assignment with expected value at least $(f(R)/2^{O(\min\{k',k-k'\})})/R^k$ times the optimum.*

Proof. The main idea of the proof is simple. We turn an instance of Max k-CSP$_R$ to an instance of Max k'-CSP$_R$ by constructing $\binom{k}{k'}R^{k-k'}$ new constraints for each original constraint; each new constraint is a projection of the original constraint to a subset of variables of size k'. We then use A to solve the newly constructed instance. Finally, B simply assigns each variable with the assignment from A with a certain probability and assigns it randomly otherwise.

For convenience, let α be $\frac{k-k'}{k}$. We define B on input $(\mathcal{X}, \mathcal{C})$ as follows:

1. Create an instance $(\mathcal{X}, \mathcal{C}')$ of Max k'-CSP$_R$ with the same variables and, for each $C = (W, S, P) \in \mathcal{C}$ and for every subset S' of S with $|S'| = k'$ and every $\tau \in [R]^{S-S'}$, create a constraint $C^{S',\tau} = (W', S', P')$ in $\mathcal{C}'$ where $W' = \frac{W}{\binom{k}{k'}R^{k-k'}}$ and $P' : [R]^{S'} \to \{0, 1\}$ is defined by

$$P'(\psi) = P(\psi \circ \tau).$$

Here $\psi \circ \tau$ is defined as follows:

$$\psi \circ \tau(x) = \begin{cases} \psi(x) & \text{if } x \in S', \\ \tau(x) & \text{otherwise.} \end{cases}$$

2. Run A on input $(\mathcal{X}, \mathcal{C}')$. Denote the output of A by φ_A.
3. For each $x \in \mathcal{X}$, with probability α, pick $\varphi_B(x)$ randomly from $[R]$. Otherwise, let $\varphi_B(x)$ be $\varphi_A(x)$.
4. Output φ_B.

We now show that φ_B has expected value at least $(f(R)/2^{O(\min\{k',k-k'\})})/R^k$ times the optimum.

First, observe that the optimum of $(\mathcal{X}, \mathcal{C}')$ is at least $1/R^{k-k'}$ times that of $(\mathcal{X}, \mathcal{C})$. To see that this is true, consider any assignment $\varphi : \mathcal{X} \to [R]$ and any constraint $C = (W, S, P)$. Its weighted contribution in $(\mathcal{X}, \mathcal{C})$ is $WP(\varphi|_S)$. On the other hand, $\frac{W}{\binom{k}{k'}R^{k-k'}}P(\varphi|_S)$ appears $\binom{k}{k'}$ times in $(\mathcal{X}, \mathcal{C}')$, once for each subset $S' \subseteq S$ of size k'. Hence, the value of φ with respect to $(\mathcal{X}, \mathcal{C}')$ is at least $1/R^{k-k'}$ times its value with respect to $(\mathcal{X}, \mathcal{C})$

Recall that the algorithm A gives an assignment of expected value at least $f(R)/R^{k'}$ times the optimum of $(\mathcal{X}, \mathcal{C}')$. Hence, the expected value of φ_A is at least $f(R)/R^k$ times the optimum of $(\mathcal{X}, \mathcal{C})$.

Next, we will compute the expected value of φ_B (with respect to $(\mathcal{X}, \mathcal{C})$). We start by computing the expected value of φ_B with respect to a fixed constraint $C = (W, S, P) \in \mathcal{C}$, i.e., $\mathbb{E}_{\varphi_B}[WP(\varphi_B|_S)]$. For each $S' \subseteq S$ of size k, we define $D_{S'}$ as the event where, in step 3, $\varphi_B(x)$ is assigned to be $\varphi_A(x)$ for all $x \in S'$ and $\varphi_B(x)$ is assigned randomly for all $x \in S - S'$.

Since $D_{S'}$ is disjoint for all $S' \subseteq S$ of size k', we have the following inequality.

$$\mathop{\mathbb{E}}_{\varphi_B}[WP(\varphi_B|_S)] \geq \sum_{\substack{S' \subseteq S \\ |S'|=k'}} \Pr[D_{S'}] \mathop{\mathbb{E}}_{\varphi_B}[WP(\varphi_B|_S) \mid D_{S'}] \tag{103}$$

$$\text{(Since } \Pr[D_{S'}] = \alpha^{k-k'}(1-\alpha)^{k'}) = \alpha^{k-k'}(1-\alpha)^{k'} \sum_{\substack{S' \subseteq S \\ |S'|=k'}} W \mathop{\mathbb{E}}_{\varphi_B}[P(\varphi_B|_S) \mid D_{S'}] \tag{104}$$

Moreover, since every vertex in $S - S'$ is randomly assigned when $D_{S'}$ occurs, $\mathbb{E}[P(\varphi_B|_S) \mid D_{S'}]$ can be view as the average value of $P((\varphi_A|_{S'}) \circ \tau)$ over all $\tau \in [R]^{S-S'}$. Hence, we can derive the following:

$$\mathop{\mathbb{E}}_{\varphi_B}[P(\varphi_B|_S) \mid D_{S'}] = \frac{1}{R^{k-k'}} \mathop{\mathbb{E}}_{\varphi_A}\left[\sum_{\tau \in [R]^{S-S'}} P((\varphi_A|_{S'}) \circ \tau)\right]. \tag{105}$$

As a result, we have

$$\mathop{\mathbb{E}}_{\varphi_B}[WP(\varphi_B|_S)] \geq \frac{\alpha^{k-k'}(1-\alpha)^{k'}}{R^{k-k'}} \left(\mathop{\mathbb{E}}_{\varphi_A}\left[\sum_{\substack{S' \subseteq S \\ |S'|=k'}} \sum_{\tau \in [R]^{S-S'}} WP((\varphi_A|_{S'}) \circ \tau)\right]\right). \tag{106}$$

By summing (106) over all constraints $C \in \mathcal{C}$, we arrive at the following inequality.

$$\mathop{\mathbb{E}}_{\varphi_B}\left[\sum_{C=(W,S,P)\in\mathcal{C}} WP(\varphi_B|_S)\right] \tag{107}$$

$$\geq \frac{\alpha^{k-k'}(1-\alpha)^{k'}}{R^{k-k'}}\mathop{\mathbb{E}}_{\varphi_A}\left[\sum_{C=(W,S,P)\in\mathcal{C}}\left(\sum_{\substack{S'\subseteq S\\|S'|=k'}}\sum_{\tau\in[R]^{S-S'}} WP((\varphi_A|_{S'})\circ\tau)\right)\right] \tag{108}$$

$$= \binom{k}{k'}\alpha^{k-k'}(1-\alpha)^{k'}\mathop{\mathbb{E}}_{\varphi_A}\left[\sum_{C=(W,S,P)\in\mathcal{C}}\left(\sum_{\substack{S'\subseteq S\\|S'|=k'}}\sum_{\tau\in[R]^{S-S'}} \frac{W}{\binom{k}{k'}R^{k-k'}}P((\varphi_A|_{S'})\circ\tau)\right)\right] \tag{109}$$

$$= \binom{k}{k'}\alpha^{k-k'}(1-\alpha)^{k'}\mathop{\mathbb{E}}_{\varphi_A}\left[\sum_{C'=(W',S',P')\in\mathcal{C}} W'P'(\varphi_A|_{S'})\right] \tag{110}$$

The first expression is the expected value of φ_B whereas the last is $\binom{k}{k'}\alpha^{k-k'}(1-\alpha)^{k'}$ times the expected value of φ_A. Since the expected value of φ_A is at least $f(R)/R^k$ times the optimum of $(\mathcal{X},\mathcal{C})$, the expected value of φ_B is at least $(\binom{k}{k'}\alpha^{k-k'}(1-\alpha)^{k'})(f(R)/R^k)$ times the optimum of $(\mathcal{X},\mathcal{C})$.

Finally, we substitute $\alpha = \frac{k-k'}{k}$ in to get

$$\binom{k}{k'}\alpha^{k-k'}(1-\alpha)^{k'} = \binom{k}{k'}\left(\frac{k-k'}{k}\right)^{k-k'}\left(\frac{k'}{k}\right)^{k'}. \tag{111}$$

Let $l = \min\{k', k-k'\}$. We then have

$$\binom{k}{k'}\left(\frac{k-k'}{k}\right)^{k-k'}\left(\frac{k'}{k}\right)^{k'} = \binom{k}{l}\left(\frac{k-l}{k}\right)^{k-l}\left(\frac{l}{k}\right)^{l} \tag{112}$$

$$\geq \left(\frac{k}{l}\right)^{l}\left(\frac{k-l}{k}\right)^{k-l}\left(\frac{l}{k}\right)^{l} \tag{113}$$

$$\geq \left(\frac{k-l}{k}\right)^{k} \tag{114}$$

$$= \left((1-l/k)^{2k/l}\right)^{2l} \tag{115}$$

$$\text{(From Bernoulli's inequality and from } l \leq k/2) \geq 1/2^{2l}. \tag{116}$$

Hence, φ_B has expected value at least $(f(R)/2^{O(l)})/R^k$ times the optimum of $(\mathcal{X},\mathcal{C})$, which completes the proof of this lemma. ◀

Finally, Theorem 3 is an immediate consequence of applying Lemma 16 to the algorithm from [24] with $k' = 2$ and $f(R) = \Omega(R\log R)$.

5 k-Query Large Alphabet Dictator Test

We remark that the results of Section 3 also implicitly yield a k-query nonadaptive *Dictator-vs.-Quasirandom* test for functions over large alphabet. A Dictator-vs.-Quasirandom test aims to distinguish dictator functions from functions with small low-degree influences ("quasirandom").

This concept was essentially introduced in [19], and we borrow the "quasirandom" terminology from [29] (adapted here for functions over non-binary alphabets). Specifically, we have the following test:

▶ **Theorem 17.** *For any function $f : [R]^n \to [R]$, and any $i \in [R]$, let $f^i : [R]^n \to \{0,1\}$ denote the indicator function for $f(x) = i$. For any $k, R \geq 2$, set parameters $\rho = 1/\sqrt{(k-1)\log R}$, $d = 10k \log R$, and $\delta = 1/(R^{10+100k\log(R)})$. Then there exists a k-query nonadaptive Dictator-vs.Quasirandom test with the following guarantees:*

Completeness: *If f is a dictator, i.e. $f(x) = x_j$ for some coordinate $j \in [n]$, then the test passes with probability at least*

$$\rho^k = 1/((\log R)^{k/2} 2^{O(k \log k)}) .$$

Soundness: *If f has $Inf_j^{\leq d}[f^i] \leq \delta$ for all coordinates $j \in [n]$ and all projections $i \in [R]$, then the test passes with probability at most*

$$2^{O(k)}/R^{k-1} .$$

Notice that if we assume f is balanced, then this theorem is immediately implied by the techniques of Section 3. However, to extend this to general functions via "folding", we must technically show that the operation of folding keeps low-influence functions as low-influence. The full proof can be found in Appendix C.

6 Conclusions and Open Questions

We conclude by posting interesting open questions regarding the approximability of Max k-CSP$_R$ and providing our opinions on each question. First, as stated earlier, even with our results, current inapproximability results do not match the best known approximation ratio achievable in polynomial time when $3 \leq k < R$. Hence, it is intriguing to ask what the right ratio that Max k-CSP$_R$ becomes NP-hard to approximate is. Since our hardness factor $2^{O(k \log k)}(\log R)^{k/2}/R^{k-1}$ does not match Chan's hardness factor $O(k/R^{k-2})$ when $k = R$, it is likely that there is a k between 3 and $R-1$ such that a drastic change in the hardness factor, and technique that yields that factor, occurs.

Moreover, since our PCP has completeness of $1/(2^{O(k)}(\log R)^{k/2})$, even if one cannot improve on the inapproximability factor, it is still interesting if one can come up with a hardness result with almost perfect completeness. In fact, even for $k = 2$, there is no known hardness of approximation of factor better than $O(\log R/\sqrt{R})$ with near perfect completeness whereas the best UGC-hardness known is $O(\log R/R)$.

It is also interesting to try to relax assumptions for other known inapproximability results from UGC to the One-Sided UGC. Since the One-Sided UGC is implied by d-to-1 Games Conjecture, doing so will imply inapproximability results based on the d-to-1 Games Conjecture. Moreover, without going into too much detail, we remark that most attempts to refute the UGC and the d-to-1 Conjecture need the value of the game to be high [1, 5, 7, 15, 20, 25, 36]. Hence, these algorithms are not candidates to refute the One-Sided UGC. In addition, Arora, Barak and Steurer's [1] subexponential time algorithm for unique games suggest that unique games have *intermediate complexity*, meaning that, even if the UGC is true, the UGC-hardness would not imply exponential time lower bounds. On the other hand, to the best of the authors' knowledge, the ABS algorithm does not run in subexponential time when the completeness is small. Hence, the One-Sided UGC may require exponential time to solve, which could give similar running time lower bounds for the resulting hardness of approximation results. Finally, there are evidences suggesting that relaxing completeness or soundness conditions of a conjecture can make it easier; the most relevant such result is that from Feige and Reichman who proved that, if one only cares about the approximation ratio and not completeness and soundness, then unique game is hard to approximate to within factor ε for any $\varepsilon > 0$ [13].

References

1 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. *J. ACM*, 62(5):42, 2015. `doi:10.1145/2775105`.

2 P. Austrin and E. Mossel. Approximation resistant predicates from pairwise independence. In *Computational Complexity, 2008. CCC'08. 23rd Annual IEEE Conference on*, pages 249–258, June 2008. `doi:10.1109/CCC.2008.20`.

3 Siu On Chan. Approximation resistance from pairwise independent subgroups. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC'13, pages 447–456, New York, NY, USA, 2013. ACM. `doi:10.1145/2488608.2488665`.

4 Moses Charikar, MohammadTaghi Hajiaghayi, and Howard Karloff. Improved approximation algorithms for label cover problems. *Algorithmica*, 61(1):190–206, 2011. `doi:10.1007/s00453-010-9464-3`.

5 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for unique games. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC'06, pages 205–214, New York, NY, USA, 2006. ACM. `doi:10.1145/1132516.1132547`.

6 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. *ACM Transactions on Algorithms*, 5(3), 2009. `doi:10.1145/1541885.1541893`.

7 Eden Chlamtac, Konstantin Makarychev, and Yury Makarychev. How to play unique games using embeddings. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 687–696, 2006. `doi:10.1109/FOCS.2006.36`.

8 Pierluigi Crescenzi, Riccardo Silvestri, and Luca Trevisan. On weighted vs unweighted versions of combinatorial optimization problems. *Information and Computation*, 167(1):10–26, 2001.

9 Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. *SIAM Journal on Computing*, 39(3):843–873, 2009. `doi:10.1137/07068062X`.

10 Irit Dinur and Igor Shinkar. On the conditional hardness of coloring a 4-colorable graph with super-constant number of colors. In *Proceedings of the 13th International Conference on Approximation, and 14 the International Conference on Randomization, and Combinatorial Optimization: Algorithms and Techniques*, APPROX/RANDOM'10, pages 138–151, Berlin, Heidelberg, 2010. Springer-Verlag. URL: `http://dl.acm.org/citation.cfm?id=1886521.1886533`.

11 Lars Engebretsen. The nonapproximability of non-boolean predicates. *SIAM J. Discret. Math.*, 18(1):114–129, January 2005. `doi:10.1137/S0895480100380458`.

12 Lars Engebretsen and Jonas Holmerin. More efficient queries in pcps for np and improved approximation hardness of maximum csp. *Random Struct. Algorithms*, 33(4):497–514, December 2008. `doi:10.1002/rsa.v33:4`.

13 Uriel Feige and Daniel Reichman. On systems of linear equations with two variables per equation. In Klaus Jansen, Sanjeev Khanna, JoséD.P. Rolim, and Dana Ron, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*,

volume 3122 of *Lecture Notes in Computer Science*, pages 117–127. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-27821-4_11.

14 Gil Goldshlager and Dana Moshkovitz. Approximating kCSP for large alphabets. https://people.csail.mit.edu/dmoshkov/papers/Approximating%20MAX%20kCSP.pdf, 2015.

15 Anupam Gupta and Kunal Talwar. Approximating unique games. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 99–106, 2006. URL: http://dl.acm.org/citation.cfm?id=1109557.1109569.

16 Venkatesan Guruswami and Prasad Raghavendra. Constraint satisfaction over a non-boolean domain: Approximation algorithms and unique-games hardness. In Ashish Goel, Klaus Jansen, JoséD.P. Rolim, and Ronitt Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, volume 5171 of *Lecture Notes in Computer Science*, pages 77–90. Springer Berlin Heidelberg, 2008. doi:10.1007/978-3-540-85363-3_7.

17 Venkatesan Guruswami and Ali Kemal Sinop. The complexity of finding independent sets in bounded degree (hyper)graphs of low chromatic number. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1615–1626, 2011. doi:10.1137/1.9781611973082.125.

18 Gustav Hast. Approximating Max kCSP – outperforming a random assignment with almost a linear factor. In *Proceedings of the 32Nd International Conference on Automata, Languages and Programming*, ICALP'05, pages 956–968, Berlin, Heidelberg, 2005. Springer-Verlag. doi:10.1007/11523468_77.

19 Johan Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 627–636. IEEE, 1996.

20 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thirty-fourth Annual ACM Symposium on Theory of Computing*, STOC'02, pages 767–775, New York, NY, USA, 2002. ACM. doi:10.1145/509907.510017.

21 Subhash Khot. On the unique games conjecture (invited survey). In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010*, pages 99–121, 2010. doi:10.1109/CCC.2010.19.

22 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.

23 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-ε. *J. Comput. Syst. Sci.*, 74(3):335–349, May 2008. doi:10.1016/j.jcss.2007.06.019.

24 Guy Kindler, Alexandra Kolla, and Luca Trevisan. Approximation of non-boolean 2csp. *CoRR*, abs/1504.00681, 2015. URL: http://arxiv.org/abs/1504.00681.

25 Alexandra Kolla. Spectral algorithms for unique games. *Computational Complexity*, 20(2):177–206, 2011. doi:10.1007/s00037-011-0011-7.

26 Konstantin Makarychev and Yury Makarychev. Approximation algorithm for non-boolean max-k-csp. *Theory of Computing*, 10:341–358, 2014. doi:10.4086/toc.2014.v010a013.

27 Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. *Ann. Math. (2)*, 171(1):295–341, 2010. doi:10.4007/annals.2010.171.295.

28 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. URL: http://www.cambridge.org/de/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/analysis-boolean-functions.

29 Ryan O'Donnell and Yi Wu. 3-bit dictator testing: 1 vs. 5/8. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 365–373, 2009. URL: `http://dl.acm.org/citation.cfm?id=1496770.1496811`.

30 Ryan O'Donnell and Yi Wu. Conditional hardness for satisfiable 3-csps. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 493–502, 2009. `doi:10.1145/1536414.1536482`.

31 Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC'08, pages 245–254, New York, NY, USA, 2008. ACM. `doi:10.1145/1374376.1374414`.

32 Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 191–199, 2000. `doi:10.1145/335305.335329`.

33 Alex Samorodnitsky and Luca Trevisan. Gowers uniformity, influence of variables, and pcps. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC'06, pages 11–20, New York, NY, USA, 2006. ACM. `doi:10.1145/1132516.1132519`.

34 Linqing Tang. Conditional hardness of approximating satisfiable max 3csp-q. In Yingfei Dong, Ding-Zhu Du, and Oscar Ibarra, editors, *Algorithms and Computation*, volume 5878 of *Lecture Notes in Computer Science*, pages 923–932. Springer Berlin Heidelberg, 2009. `doi:10.1007/978-3-642-10631-6_93`.

35 Luca Trevisan. Parallel approximation algorithms by positive linear programming. *Algorithmica*, 21(1):72–88, 1998. `doi:10.1007/PL00009209`.

36 Luca Trevisan. Approximation algorithms for unique games. *Theory of Computing*, 4(1):111–128, 2008. `doi:10.4086/toc.2008.v004a005`.

A Proofs of Preliminary Results

For completeness, we prove some of the preliminary results, whose formal proofs were not found in the literature by the authors.

A.1 Mollification Lemma

Below is the proof of the Mollification Lemma. We remark that, while its main idea is explained in [28], the full proof is not shown there. Hence, we provide the proof here for completeness.

Proof of Lemma 10. Let $p : \mathbb{R} \to \mathbb{R}$ be a $\mathcal{C}^4$ function supported only on $[-1, +1]$, such that $p(y)$ forms a probability distribution. (For example, an appropriately normalized version of $e^{-1/(1+y^2)}$ for $|y| \leq 1$). Define $p_\lambda(y)$ to be re-scaled to have support $[-\lambda, +\lambda]$ for some $\lambda > 0$:

$$p_\lambda(y) := (1/\lambda)p(y/\lambda). \tag{117}$$

Let Y_λ be a random variable with distribution $p_\lambda(y)$, supported on $[-\lambda, +\lambda]$. We will set $\lambda = \zeta/c$.

Now, define

$$\widetilde{\psi} := \mathop{\mathbb{E}}_{Y_\lambda}[\psi(x + Y_\lambda)]. \tag{118}$$

This is pointwise close to ψ, since ψ is c-Lipschitz:

$$|\widetilde{\psi}(x)-\psi(x)| = |\mathop{\mathbb{E}}_{Y_\lambda}[\psi(x+Y_\lambda)-\psi(x)]| \leq \mathop{\mathbb{E}}_{Y_\lambda}[|\psi(x+Y_\lambda)-\psi(x)|] \leq \mathop{\mathbb{E}}_{Y_\lambda}[c|Y_\lambda|] \leq c\lambda = \zeta. \tag{119}$$

Further, $\widetilde{\psi}$ is $\mathcal{C}^3$, because $\widetilde{\psi}(x)$ can be written as a convolution:

$$\widetilde{\psi}(x) = (\psi * p_\lambda)(x) \implies \widetilde{\psi}''' = (\psi * p_\lambda)''' = (\psi * p_\lambda'''). \tag{120}$$

To see that $\widetilde{\psi}'''$ is bounded, for a fixed $x \in \mathbb{R}$, define the constant $z := \psi(x)$. Then,

$$\begin{aligned}
|\widetilde{\psi}'''(x)| &= |(\psi * p_\lambda''')(x)| && (121)\\
(z \text{ is constant, so } z'=0) \quad &= |(\psi * p_\lambda''' - z' * p_\lambda'')(x)| && (122)\\
&= |(\psi * p_\lambda''' - z * p_\lambda''')(x)| && (123)\\
&= |((\psi - z) * p_\lambda''')(x)| && (124)\\
&= \left|\int_{-\infty}^{+\infty} p_\lambda'''(y)(\psi(x-y)-z)dy\right| && (125)\\
&= \left|\int_{-\infty}^{+\infty} p_\lambda'''(y)(\psi(x-y)-\psi(x))dy\right| && (126)\\
&\leq \int_{-\lambda}^{+\lambda} |p_\lambda'''(y)||\psi(x-y)-\psi(x)|dy && (127)\\
(c\text{-Lipschitz}) \quad &\leq ||p_\lambda'''||_\infty \int_{-\lambda}^{+\lambda} |cy|dy && (128)\\
&= ||p_\lambda'''||_\infty c\lambda^2. && (129)
\end{aligned}$$

Define the universal constant $\widetilde{C} := ||p'''||_\infty$. We have

$$p_\lambda'''(y) = (1/\lambda^4)p'''(y/\lambda) \implies ||p_\lambda'''||_\infty \leq (1/\lambda^4)\widetilde{C}. \tag{130}$$

With our choice of $\lambda = \zeta/c$, this yields $|\widetilde{\psi}'''(x)| \leq \widetilde{C}c^3/\zeta^2$, which completes the proof of Lemma 10. ◀

A.2 Proof of Lemma 11

Below we show the proof of Lemma 11.

Proof. First, we "mollify" the function ψ to construct a $\mathcal{C}^3$ function $\widetilde{\psi}$, by applying Lemma 10 for $\zeta = 1/R^k$. Notice that both choices of ψ are k-Lipschitz. Therefore the Mollification Lemma guarantees that $|\widetilde{\psi}'''(x)| \leq \widetilde{C}k^3R^{2k}$ for some universal constant $\widetilde{C}$.

Since $\widetilde{\psi}$ is pointwise close to ψ, with deviation at most $1/R^k$, we have

$$\begin{aligned}
&\left|\mathop{\mathbb{E}}_{y\in\{\pm1\}^{nR}}[\psi(F^{\leq d}(y))] - \mathop{\mathbb{E}}_{x\in[R]^n}[\psi(f^{\leq d}(x))]\right|\\
\leq &\left|\mathop{\mathbb{E}}_{y\in\{\pm1\}^{nR}}[\widetilde{\psi}(F^{\leq d}(y))] - \mathop{\mathbb{E}}_{x\in[R]^n}[\widetilde{\psi}(f^{\leq d}(x))]\right| + O(1/R^k).
\end{aligned} \tag{131}$$

Applying the General Invariance Principle (Theorem 9) with the function $\widetilde{\psi}$, we have

$$\left|\mathop{\mathbb{E}}_{y\in\{\pm1\}^{nR}}[\widetilde{\psi}(F^{\leq d}(y))] - \mathop{\mathbb{E}}_{x\in[R]^n}[\widetilde{\psi}(f^{\leq d}(x))]\right| \leq \widetilde{C}k^3R^{2k}10^dR^{d/2}\sqrt{\delta}. \tag{132}$$

By our choice of parameters d, δ, this is $O(1/R^k)$. ◀

B d-to-1 Games Conjecture implies One-Sided Unique Games Conjecture

In this section, we prove that if d-to-1 Games Conjecture is true, then so is One-Sided Unique Games Conjecture.

▶ **Lemma 18.** *For every $d \in \mathbb{N}$, d-to-1 Games Conjecture implies One-Sided UGC.*

Proof. Suppose that d-to-1 Games Conjecture is true for some $d \in \mathbb{N}$. We will prove One-Sided UGC; more specifically, ζ in the One-Sided UGC is $1/d$. The reduction from a d-to-1 game $(V, W, E, N, \{\pi_e\}_{e \in E})$ to a unique game $(V', W', E', N', \{\pi'_e\}_{e \in E})$ can be described as follows:

- Let $V' = V, W' = W, E' = E$, and $N' = N$
- We define π'_e as follows. For each $\theta \in [N/d]$, let the elements of $\pi_e^{-1}(\theta)$ be $\sigma_1, \sigma_2, \dots, \sigma_d \in [N]$. We then define $\pi'_e(\sigma_i) = d(\theta - 1) + i$.

Now, we will prove the soundness and completeness of this reduction.

Completeness. Suppose that the d-to-1 game is satisfiable. Let $\varphi : V \cup W \to [N]$ be the assignment that satisfies every constraint in the d-to-1 game. We define $\varphi' : V' \cup W' \to [N']$ by first assign $\varphi'(v) = \varphi(v)$ for every $v \in V$. Then, for each $w \in W$, pick $\varphi'(w)$ to be an assignment that satisfies as many edges touching w in the unique game as possible, i.e., for a fixed w, $\varphi'(w)$ is select to maximize $|\{v \in N(w) \mid \pi'_{(v,w)}(\varphi(v)) = \varphi'(w)\}|$ where $N(w)$ is the set of neighbors of w. From how $\varphi'(w)$ is picked, we have

$$\begin{aligned}&|\{v \in N(w) \mid \pi'_{(v,w)}(\varphi(v)) = \varphi'(w)\}| \\ \geq &\frac{1}{d}\sum_{i=1}^{d} |\{v \in N(w) \mid \pi'_{(v,w)}(\varphi(v)) = d(\varphi(w) - 1) + i\}|. \end{aligned} \tag{133}$$

Let $1[\pi'_{(v,w)}(\varphi(v)) = d(\varphi(w) - 1) + i]$ be the indicating variable whether $\pi'_{(v,w)}(\varphi(v)) = d(\varphi(w) - 1) + i$, we can rewrite the right hand side as follows:

$$\frac{1}{d}\sum_{i=1}^{d} |\{v \in N(w) \mid \pi'_{(v,w)}(\varphi(v)) = d(\varphi(w) - 1) + i\}| \tag{134}$$

$$= \frac{1}{d}\sum_{i=1}^{d}\sum_{v \in N(w)} 1[\pi'_{(v,w)}(\varphi(v)) = d(\varphi(w) - 1) + i] \tag{135}$$

$$= \frac{1}{d}\sum_{v \in N(w)}\sum_{i=1}^{d} 1[\pi'_{(v,w)}(\varphi(v)) = d(\varphi(w) - 1) + i]. \tag{136}$$

From how $\pi'_{(v,w)}$ is defined and since $\pi_{(v,w)}(\varphi(v)) = \varphi(w)$, there exists $i \in [d]$ such that $\pi'_{(v,w)}(\varphi(v)) = d(\varphi(w) - 1) + i$. As a result, we have

$$\frac{1}{d}\sum_{v \in N(w)}\sum_{i=1}^{d} 1[\pi'_{(v,w)}(\varphi(v)) = d(\varphi(w) - 1) + i] \geq \frac{1}{d}\sum_{v \in N(w)} 1 = \frac{|N(w)|}{d}. \tag{137}$$

In other words, at least $1/d$ fraction of edges touching w is satisfied in the unique game for every $w \in W$. Hence, φ' has value at least $1/d$, which means that the unique game also has value at least $1/d$.

Soundness. Suppose that the value of the d-to-1 game is at most γ. For any assignment $\varphi' : V' \cup W' \to [N']$ to the unique game, we can define an assignment $\varphi : V \cup W \to [N]$ by

$$\varphi(u) = \begin{cases} \varphi'(u) & \text{if } u \in V, \\ \lfloor (\varphi'(u) - 1)/d \rfloor + 1 & \text{if } u \in W. \end{cases} \tag{138}$$

From how π'_e is defined, it is easy to see that, if $\pi'_e(\varphi'(v)) = \varphi'(w)$, then $\pi_e(\varphi(v)) = \varphi(w)$. In other words, the value of φ' with respect to the unique game is no more than the value of φ with respect to the d-to-1 game. As a result, the value of the unique game is at most ε.

As a result, if it is NP-hard to distinguish a satisfiable d-to-1 game from one with value at most γ, then it is also NP-hard to distinguish a unique game of value at least $\zeta = 1/d$ from that with value at most γ, which concludes the proof of this lemma. ◀

C Proof of Dictator Test

Here we prove our result for the Dictator-vs.-Quasirandom test (Theorem 17).

Proof of Theorem 17. For $c \in [R]$, define the function

$$f_c(x_1, x_2, \dots, x_n) := f(x_1 + c, x_2 + c, \dots, x_n + c) - c. \tag{139}$$

Note that $\pm c$ is performed modulo R.

The test works as follows: Pick $z \in [R]^n$ uniformly at random, and let $x^{(1)}, x^{(2)}, \dots, x^{(k)}$ be independent ρ-correlated noisy copies of z. Then, pick $c_1, c_2, \dots, c_k$ independently uniformly at random, where each $c_i \in [R]$. Accept iff

$$f_{c_1}(x^{(1)}) = f_{c_2}(x^{(2)}) = \dots = f_{c_k}(x^{(k)}). \tag{140}$$

For completeness, notice that if f is a dictator, then $f_c = f$ for all $c \in [R]$. Say f is a dictator on the j-th coordinate: $f(x) = x_j$. Then the test clearly accepts with probability at least ρ^k (if none of the coordinates j were perturbed in all the noisy copies $x^{(i)} \overset{\rho}{\leftarrow} z$).

For soundness: For any $i \in [R]$, let $f^i : [R]^n \to \{0, 1\}$ denote the indicator function for $f(x) = i$, and similarly for $f^i_c : [R]^n \to \{0, 1\}$. Notice that

$$f^i_c(x) = f^{i+c}(x + (c, c, \dots, c)) \tag{141}$$

Then, write the acceptance probability as

$$\begin{aligned} \Pr[accept] &= \Pr_{c_i, z, x^{(j)} \overset{\rho}{\leftarrow} z} [f_{c_1}(x^{(1)}) = f_{c_2}(x^{(2)}) = \dots = f_{c_k}(x^{(k)})] && (142) \\ &= \sum_{i \in [R]} \Pr_{c_i, z, x^{(j)} \overset{\rho}{\leftarrow} z} [i = f_{c_1}(x^{(1)}) = f_{c_2}(x^{(2)}) = \dots = f_{c_k}(x^{(k)})] && (143) \\ &= \sum_{i \in [R]} \mathop{\mathbb{E}}_{c_i, z, x^{(j)} \overset{\rho}{\leftarrow} z} [f^i_{c_1}(x^{(1)}) f^i_{c_2}(x^{(2)}) \dots f^i_{c_k}(x^{(k)})] && (144) \\ \text{(Independence of } c_i) \quad &= \sum_{i \in [R]} \mathop{\mathbb{E}}_{z, x^{(j)} \overset{\rho}{\leftarrow} z} [\mathop{\mathbb{E}}_{c_1}[f^i_{c_1}(x^{(1)})] \mathop{\mathbb{E}}_{c_2}[f^i_{c_2}(x^{(2)})] \dots \mathop{\mathbb{E}}_{c_2}[f^i_{c_k}(x^{(k)})]]. && (145) \\ & && (146) \end{aligned}$$

If we define the function $g^i : [R]^n \to [0, 1]$ as

$$g^i(x) := \mathop{\mathbb{E}}_{c}[f^i_c(x)]. \tag{147}$$

Then this acceptance probability is

$$\Pr[accept] = \sum_{i\in[R]} \mathop{\mathbb{E}}_{z, x_i \overset{\rho}{\leftarrow} z} [g^i(x^{(1)})g^i(x^{(2)})\dots g^i(x^{(k)})] \tag{148}$$

$$= \sum_{i\in[R]} \mathop{\mathbb{E}}_{z}[(T_\rho g^i(z))^k]. \tag{149}$$

Notice that $\mathbb{E}_x[g^i(x)] = 1/R$, because

$$\mathop{\mathbb{E}}_{x}[g^i(x)] = \mathop{\mathbb{E}}_{x,c}[f_c^i(x)] = \mathop{\mathbb{E}}_{x,c}[f^{i+c}(x + (c, c, \dots, c))] \tag{150}$$

$$(c, x \text{ same joint distribution as } i + c, x + c) \quad = \mathop{\mathbb{E}}_{x,c}[f^c(x)] \tag{151}$$

$$= \mathop{\mathbb{E}}_{x}[\mathop{\mathbb{E}}_{c}[f^c(x)]] = \mathop{\mathbb{E}}_{x}[1/R] = 1/R. \tag{152}$$

Thus, if the function g^i has small low-degree influences, then Lemma 15 (Main Lemma) applied to g^i in line (149) directly implies that this acceptance probability is $2^{O(k)}/R^{k-1}$. We will now formally show that the influences of the "expected folded function" g^i are bounded by the influences of the original f^i.

First, the Fourier coefficients of g^i are

$$\hat{g}^i(s) = \mathop{\mathbb{E}}_{c}[\hat{f}_c^i(s)]. \tag{153}$$

Thus the low-degree influences of g^i are bounded as

$$Inf_j^{\le d}[g^i] = \sum_{\substack{s\in[R]^n \\ s(j)\ne 1, |s|\le d}} \hat{g}^i(s)^2 \tag{154}$$

$$= \sum_{\substack{s\in[R]^n \\ s(j)\ne 1, |s|\le d}} \mathop{\mathbb{E}}_{c}[\hat{f}_c^i(s)]^2 \tag{155}$$

$$\le \sum_{\substack{s\in[R]^n \\ s(j)\ne 1, |s|\le d}} \mathop{\mathbb{E}}_{c}[\hat{f}_c^i(s)^2] \tag{156}$$

$$= \mathop{\mathbb{E}}_{c}[\sum_{\substack{s\in[R]^n \\ s(j)\ne 1, |s|\le d}} \hat{f}_c^i(s)^2] \tag{157}$$

$$= \mathop{\mathbb{E}}_{c}[Inf_j^{\le d}[f_c^i]]. \tag{158}$$

Finally, we must relate the influences of f_c^i to the influences of f^i. For a fixed $c \in [R]$, we have

$$Inf_j^{\le d}[f_c^i] = Inf_j[(f_c^i)^{\le d}] \tag{159}$$

$$= \mathop{\mathbb{E}}_{x\in[R]^n}[Var_{x_j\in[R]}[(f_c^i)^{\le d}]] \tag{160}$$

$$= \mathop{\mathbb{E}}_{x\in[R]^n}[Var_{x_j\in[R]}[(f^{i+c})^{\le d}(x_1 + c, x_2 + c, \dots, x_n + c)]] \tag{161}$$

$$= \mathop{\mathbb{E}}_{x\in[R]^n}[Var_{x_j\in[R]}[(f^{i+c})^{\le d}(x_1, x_2, \dots, x_n)]] \tag{162}$$

$$= Inf_j[(f^{i+c})^{\le d}] \tag{163}$$

$$= Inf_j^{\le d}[f^{i+c}]. \tag{164}$$

Therefore, if $Inf_j^{\leq d}[f^i] \leq \delta$ for all coordinates $j \in [n]$ and all projections $i \in [R]$ (as we assume for soundness), then from (158) and (164) we have

$$Inf_j^{\leq d}[g^i] \leq \mathop{\mathbb{E}}_c[Inf_j^{\leq d}[f_c^i]] = \mathop{\mathbb{E}}_c[Inf_j^{\leq d}[f^{i+c}]] \leq \delta. \tag{165}$$

Thus the function g^i has small low-degree influences as well.

So we can complete the proof, continuing from line (149) and applying our Main Lemma to g^i:

$$\begin{aligned} \Pr[accept] &= \sum_{i\in[R]} \mathop{\mathbb{E}}_z[(T_\rho g^i(z))^k] && (166)\\ \text{(Lemma 15)} \quad &\leq \sum_{i\in[R]} 2^{O(k)}/R^k && (167)\\ &= 2^{O(k)}/R^{k-1}. && (168)\end{aligned}$$

◀

Constant-Factor Approximations for Asymmetric TSP on Nearly-Embeddable Graphs*

Dániel Marx[1], Ario Salmasi[2], and Anastasios Sidiropoulos[3]

1 **Institute of Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
`dmarx@cs.bme.hu`

2 **Dept. of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA**
`salmasi.1@osu.edu`

3 **Dept. of Computer Science and Engineering and Dept. of Mathematics, The Ohio State University, Columbus, OH, USA**
`sidiropoulos.1@osu.edu`

Abstract

In the Asymmetric Traveling Salesperson Problem (ATSP) the goal is to find a closed walk of minimum cost in a directed graph visiting every vertex. We consider the approximability of ATSP on topologically restricted graphs. It has been shown by Oveis Gharan and Saberi [13] that there exists polynomial-time constant-factor approximations on planar graphs and more generally graphs of constant orientable genus. This result was extended to non-orientable genus by Erickson and Sidiropoulos [8].

We show that for any class of *nearly-embeddable* graphs, ATSP admits a polynomial-time constant-factor approximation. More precisely, we show that for any fixed $k \geq 0$, there exist $\alpha, \beta > 0$, such that ATSP on n-vertex k-nearly-embeddable graphs admits an α-approximation in time $O(n^\beta)$. The class of k-nearly-embeddable graphs contains graphs with at most k apices, k vortices of width at most k, and an underlying surface of either orientable or non-orientable genus at most k. Prior to our work, even the case of graphs with a single apex was open. Our algorithm combines tools from rounding the Held-Karp LP via thin trees with dynamic programming.

We complement our upper bounds by showing that solving ATSP exactly on graphs of pathwidth k (and hence on k-nearly embeddable graphs) requires time $n^{\Omega(k)}$, assuming the Exponential-Time Hypothesis (ETH). This is surprising in light of the fact that both TSP on undirected graphs and Minimum Cost Hamiltonian Cycle on directed graphs are FPT parameterized by treewidth.

1998 ACM Subject Classification F.2.2 [Analysis of Algorithms and Problem Complexity] Non-numerical Algorithms and Problems–Computations on discrete structures, G.2.2 [Discrete Mathematics] Graph Theory–Graph algorithms, Path and circuit problems

Keywords and phrases asymmetric TSP, approximation algorithms, nearly-embeddable graphs, Held-Karp LP, exponential time hypothesis

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.16

* Research supported by the European Research Council (ERC) grant "PARAMTIGHT: Parameterized complexity and the search for tight complexity results," reference 280152 and OTKA grant NK105645, and by the National Science Foundation (NSF) under grant CCF 1423230 and award CAREER 1453472.

1 Introduction

An instance of the Asymmetric Traveling Salesman Problem (ATSP) consists of a directed graph $\vec{G}$ and a (not necessarily symmetric) cost function $c : E(\vec{G}) \to \mathbb{R}^+$. The goal is to find a spanning closed walk of $\vec{G}$ with minimum total cost. This is one of the most well-studied NP-hard problems.

Asadpour *et al.* [2] obtained a polynomial-time $O(\log n/ \log\log n)$-approximation algorithm for ATSP, which was the first asymptotic improvement in almost 30 years [12, 3, 9, 18]. Building on their techniques, Oveis Gharan and Saberi [13] described a polynomial-time $O(\sqrt{g}\log g)$-approximation algorithm when the input includes an embedding of the input graph into an orientable surface of genus g. Erickson and Sidiropoulos [8] improved the dependence on the genus by obtaining a $O(\log g/ \log\log g)$-approximation.

Anari and Oveis Gharan [1] have recently shown that the integrality gap of the natural linear programming relaxation of ATSP proposed by Held and Karp [16] is $\log\log^{O(1)} n$. This implies a polynomial-time $\log\log^{O(1)} n$-approximation algorithm for the *value* of ATSP. We remark that the best known lower bound on the integrality gap of the Held-Karp LP is 2 [4]. Obtaining a polynomial-time constant-factor approximation algorithm for ATSP is a central open problem in optimization.

1.1 Our contribution

We study the approximability of ATSP on topologically restricted graphs. Prior to our work, a constant-factor approximation algorithm was known only for graphs of bounded genus. We significantly extend this result by showing that there exists a polynomial-time constant-factor approximation algorithm for ATSP on nearly embeddable graphs. These graphs include graphs with bounded genus, with a bounded number of apices and a bounded number of vortices of bounded pathwidth. We remark that prior to our work, even the case of planar graphs with a single apex was open[1]. For any $a, g, k, p \geq 0$, we say that a graph is (a, g, k, p)-nearly embeddable if it is obtained from a graph of Euler genus g by adding a apices and k vortices of pathwidth p (see [20, 19, 7] for more precise definitions). The following summarizes our result.

▶ **Theorem 1.** *Let $a, g, k \geq 0$, $p \geq 1$. There is a $O(a(g+k+1)+p^2)$-approximation algorithm for ATSP on (a, g, k, p)-nearly embeddable digraphs, with running time $n^{O((a+p)(g+k+1)^4)}$.*

The above algorithm is obtained via a new technique that combines the Held-Karp LP with a dynamic program that solves the problem on vortices. We remark that it is not known whether the integrality gap of the LP is constant for graphs of constant pathwidth.

We complement this result by showing that solving ATSP exactly on graphs of pathwidth p (and hence on p-nearly embeddable graphs) requires time $n^{\Omega(p)}$, assuming the Exponential-Time Hypothesis (ETH). This is surprising in light of the fact that both TSP on undirected graphs and Minimum Cost Hamiltonian Cycle on directed graphs are FPT parameterized by treewidth. The following summarizes our lower bound.

[1] Previous algorithms for constructing thin trees [13] and forests [8] on surface-embedded graphs depend critically on the relation between cuts and cycles in the dual graph, and thus are not directly applicable to the case of graphs even with a single apex. We also remark that the optimal walk might traverse the apex arbitrarily many times; thus, any approach that attempts to first solve the problem on the subgraph obtained by removing the apex, cannot yield a constant-factor approximation.

▶ **Theorem 2.** *Assuming ETH, there is no $f(p)n^{o(p)}$ time algorithm for ATSP on graphs of pathwidth at most p for any computable function f.*

1.2 Overview of the algorithm

We now give a high level overview of the main steps of the algorithm and highlight some of the main challenges.

Step 1: Reducing the number of vortices. We first reduce the problem to the case of nearly embeddable graphs with a single vortex. This is done by iteratively merging pairs of vortices. We can merge two vortices by adding a new handle on the underlying surface-embedded graph. For the remainder we will focus on the case of graphs with a single vortex.

Step 2: Traversing a vortex. We obtain an exact polynomial-time algorithm for computing a closed walk that visits all the vertices in the vortex. We remark that this subsumes as a special case the problem of visiting all the vertices in a single face of a planar graph, which was open prior to our work.

Let us first consider the case of a vortex in a planar graph. Let $\vec{W}$ be an optimal walk that visits all the vertices in the vortex. Let F be the face on which the vortex is attached. We give a dynamic program that maintains a set of partial solutions for each subpath of F. We prove correctness of the algorithm by establishing structural properties of $\vec{W}$. The main technical difficulty is that $\vec{W}$ might be self-crossing. We first decompose $\vec{W}$ into a collection $\mathcal{W}$ of non-crossing walks. We form a conflict graph $\mathcal{I}$ of $\mathcal{W}$ and consider a spanning forest $\mathcal{F}$ of $\mathcal{I}$. This allows us to prove correctness via induction on the trees of $\mathcal{F}$.

The above algorithm can be extended to graphs of bounded genus. The main difference is that the dynamic program now computes a set of partial solutions for each bounded collection of subpaths of F.

Finally, the algorithm is extended to the case of nearly-embeddable graphs by adding the apices to the vortex without changing the cost of the optimum walk.

Step 3: Finding a thin forest in the absence of vortices. The constant-factor approximation for graphs of bounded genus was obtained by constructing thin forests with a bounded number of components in these graphs [13, 8]. We extend this result by constructing thin forests with a bounded number of components in graphs of bounded genus and with a bounded number of additional apices. Prior to our work even the case of planar graphs with a single apex was open; in fact, no constant-factor approximation algorithm was known for these graphs.

Step 4: Combining the Held-Karp LP with the dynamic program. We next combine the dynamic program with the thin forest construction. We first compute an optimal walk $\vec{W}$ visiting all the vertices in the vortex, and we contract the vortex into a single vertex. A natural approach would be to compute a thin forest in the contracted graph. Unfortunately this fails because such a forest might not be thin in the original graph. In order to overcome this obstacle we change the feasible solution of the Held-Karp LP by taking into account $\vec{W}$, and we modify the forest construction so that it outputs a subgraph that is thin with respect to this new feasible solution.

Step 5: Rounding the forest into a walk. Once we have a thin spanning subgraph of G we can compute a solution to ATSP via circulations, as in previous work.

1.3 Organization

The rest of the paper is organized as follows. Section 2 introduces some basic notation. Section 3 defines the Held-Karp LP for ATSP. Section 4 presents the main algorithm, using the dynamic program and the thin forest construction as a black box. Section 5 presents the technique for combining the dynamic program with the Held-Karp LP. Section 6 gives the algorithm for computing a thin tree in a 1-apex graph. This algorithm is generalized to graphs with a bounded number of apices in Section 7, and to graphs of bounded genus and with a bounded number of apices in Section 8. In Section 9 we show how to modify the thin forest construction so that we can compute a spanning thin subgraph in a nearly-embeddable graph, using the solution of the dynamic program.

The dynamic program is given in Sections 10, 11, 12, 13, and 14. More precisely, Section 10 introduces a certain preprocessing step. Section 11 establishes a structural property of the optimal solution. Section 12 presents the dynamic program for a vertex in a planar graph. Sections 13 and 14 generalize this dynamic program to graphs of bounded genus and with a bounded number of apices respectively.

Finally, Section 15 presents the lower bound.

2 Notation

In this section we introduce some basic notation that will be used throughout the paper.

Graphs. Unless otherwise specified, we will assume that for every pair of vertices in a graph there exists a unique shortest path; this property can always be achieved by breaking ties between different shortest paths in a consistent manner (e.g. lexicographically). Moreover for every edge of a graph (either directed or undirected) we will assume that its length is equal to the shortest path distance between its endpoints. Let $\vec{G}$ be some digraph. Let G be the undirected graph obtained from $\vec{G}$ by ignoring the directions of the edges, that is $V(G) = V(\vec{G})$ and $E(G) = \{\{u,v\} : (u,v) \in E(\vec{G}) \text{ or } (v,u) \in E(\vec{G})\}$. We say that G is the *symmetrization* of $\vec{G}$. For some $x : E(\vec{G}) \to \mathbb{R}$ we define $\mathsf{cost}_{\vec{G}}(x) = \sum_{(u,v)\in E(\vec{G})} x((u,v)) \cdot d_{\vec{G}}(u,v)$. For a subgraph $S \subseteq G$ we define $\mathsf{cost}_G(S) = \sum_{e\in E(S)} c(e)$. Let z be a weight function on the edges of G. For any $A, B \subseteq V(G)$ we define $z(A,B) = \sum_{a\in A, b\in B} z(\{a,b\})$.

Asymmetric TSP. Let $\vec{G}$ be a directed graph with non-negative arc costs. For each arc $(u,v) \in E(\vec{G})$ we denote the cost of (u,v) by $c(u,v)$. A *tour* in $\vec{G}$ is a closed walk in $\vec{G}$. The *cost* of a tour $\tau = v_1, v_2, \ldots, v_k, v_1$ is defined to be $\mathsf{cost}_{\vec{G}}(\tau) = d_{\vec{G}}(v_k, v_1) + \sum_{i=1}^{k-1} d_{\vec{G}}(v_i, v_{i+1})$. Similarly the cost of an open walk $W = v_1, \ldots, v_k$ is defined to be $\mathsf{cost}_{\vec{G}}(W) = \sum_{i=1}^{k-1} d_{\vec{G}}(v_i, v_{i+1})$. The cost of a collection $\mathcal{W}$ of walks is defined to be $\mathsf{cost}_{\vec{G}}(\mathcal{W}) = \sum_{W\in\mathcal{W}} \mathsf{cost}_{\vec{G}}(W)$. We denote by $\mathsf{OPT}_{\vec{G}}$ the minimum cost of a tour traversing all vertices in $\vec{G}$. For some $U \subseteq V(\vec{G})$ we denote by $\mathsf{OPT}_{\vec{G}}(U)$ the minimum cost of a tour in $\vec{G}$ that visits all vertices in U.

3 The Held-Karp LP

We recall the Held-Karp LP for ATSP [15]. Fix a directed graph $\vec{G}$ and a cost function $c : E(\vec{G}) \to \mathbb{R}^+$. For any subset $U \subseteq V$, we define $\delta^+_{\vec{G}}(U) := \{(u,v) \in E(\vec{G}) : u \in U \text{ and } v \notin U\}$ and $\delta^-_{\vec{G}}(U) := \delta^+_{\vec{G}}(V \setminus U)$. We omit the subscript $\vec{G}$ when the underlying graph is clear from context. We also write $\delta^+(v) = \delta^+(\{v\})$ and $\delta^-(v) = \delta^-(\{v\})$ for any single vertex v.

Let G be the symmetrization of $\vec{G}$. For any $U \subseteq V(G)$, we define $\delta_G(U) := \{\{u,v\} \in E(G) : u \in U \text{ and } v \notin U\}$. Again, we omit the subscript G when the underlying graph is clear from context. We also extend the cost function c to undirected edges by defining $c(\{u,v\}) := \min\{c((u,v)), c((v,u))\}$. For any function $x\colon E(\vec{G}) \to \mathbb{R}$ and any subset $W \subseteq E(\vec{G})$, we write $x(W) = \sum_{a\in W} x(a)$. With this notation, the Held-Karp LP relaxation is defined as follows.

$$\boxed{\begin{array}{lrl} \text{minimize} & \multicolumn{2}{l}{\sum_{a\in E(\vec{G})} c(a)\cdot x(a)} \\ \text{subject to} & x(\delta^+(U)) \geq 1 & \text{for all nonempty } U \subsetneq V(\vec{G}) \\ & x(\delta^+(v)) = x(\delta^-(v)) & \text{for all } v \in V(\vec{G}) \\ & x(a) \geq 0 & \text{for all } a \in E(\vec{G}) \end{array}}$$

We define the *symmetrization* of x as the function $z\colon E(G) \to \mathbb{R}$ where $z(\{u,v\}) := x((u,v)) + x((v,u))$ for every edge $\{u,v\} \in E(G)$. For any subset $W \subseteq E(G)$ of edges, we write $z(W) := \sum_{e\in W} z(e)$. Let $\vec{W} \subseteq \vec{G}$. Let $\alpha, s > 0$. We say that $\vec{W}$ is α-*thin* (w.r.t. z) if for all $U \subseteq V$ we have $|E(\vec{W}) \cap \delta(U)| \leq \alpha \cdot z(\delta(U))$. We also say that $\vec{W}$ is (α, s)-*thin* (w.r.t. x) if $\vec{W}$ is α-thin (w.r.t. z) and $c(E(\vec{W})) \leq s \cdot \sum_{e\in E(\vec{G})} c(e)\cdot x(e)$. We say that z is $\vec{W}$-*dense* if for all $(u,v) \in E(\vec{W})$ we have $z(\{u,v\}) \geq 1$. We say that z is ε-*thick* if for all $U \subsetneq V(G)$ with $U \neq \emptyset$ we have $z(\delta(U)) \geq \varepsilon$.

4 An approximation algorithm for nearly-embeddable graphs

The following Lemma is implicit in the work of Erickson and Sidiropoulos [8] (see also [2]).

▶ **Lemma 3.** *Let $\vec{G}$ be a digraph and let x be a feasible solution for the Held-Karp LP for $\vec{G}$. Let $\alpha, s > 0$, and let S be a (α, s)-thin spanning subgraph of G (w.r.t. x), with at most k connected components. Then, there exists a polynomial-time algorithm which computes a collection of closed walks $C_1, \ldots, C_{k'}$, for some $k' \leq k$, such that their union visits all the vertices in $V(\vec{G})$, and such that $\sum_{i=1}^{k} \mathsf{cost}_{\vec{G}}(C_i) \leq (2\alpha + s) \sum_{e\in E(\vec{G})} c(e)\cdot x(e)$.*

The following is the main technical Lemma that combines a solution to the Held-Karp LP with a walk traversing the vortex that is computed via the dynamic program. The proof of Lemma 4 is deferred to Section 5. A similar result, for the case of graphs of orientable genus, was first obtained in [13].

▶ **Lemma 4.** *Let $a, g, p > 0$, let $\vec{G}$ be a $(a, g, 1, p)$-nearly embeddable graph, and let G be its symmetrization. There exists an algorithm with running time $n^{O((a+p)g^4)}$ which computes a feasible solution x for the Held-Karp LP for $\vec{G}$ with cost $O(\mathsf{OPT}_{\vec{G}})$ and a spanning subgraph S of G with at most $O(a + g)$ connected components, such that S is $(O(a\cdot g + p^2), O(1))$-thin w.r.t. x.*

Using Lemma 4 we are now ready to obtain an approximation algorithm for nearly-embeddable graphs with a single vortex.

▶ **Theorem 5.** *Let $a, g \geq 0$, $p \geq 1$. There exists a $O(a\cdot g + p^2)$-approximation algorithm for ATSP on $(a, g, 1, p)$-nearly embeddable digraphs, with running time $n^{O((a+p)(g+1)^4)}$.*

Proof. We follow a similar approach to [8]. The only difference is that in [8] the algorithm uses an optimal solution to the Held-Karp LP. In contrast, here we use a feasible solution that is obtained by Lemma 4, together with an appropriate thin subgraph.

Let $\vec{G}$ be $(a, g, 1, p)$-nearly embeddable digraph. By using Lemma 4, we find in time $n^{O((a+p)g^4)}$ a feasible solution x for the Held-Karp LP for $\vec{G}$ with cost $O(\mathsf{OPT}_{\vec{G}})$ and a spanning subgraph S of G with at most $O(a + g)$ connected components, such that S is $(O(a \cdot g + p^2), O(1))$-thin w.r.t. x. Now we compute in polynomial time a collection of closed walks $C_1, \ldots, C_{k'}$, for some $k' \in O(a + g)$, that visit all the vertices in $V(\vec{G})$, and such that the total cost of all walks is at most $O((a \cdot g + p^2) \cdot OPT_{\vec{G}})$, using Lemma 3. For every $i \in \{1, \ldots, k'\}$, let $v_i \in V(\vec{G})$ be an arbitrary vertex visited by C_i. We construct a new instance $(\vec{G}', c')$ of ATSP as follows. Let $V(\vec{G}') = \{v_1, \ldots, v_{k'}\}$. For any $u, v \in V(\vec{G}')$, we have an edge (u, v) in $E(\vec{G}')$, with $c'(u, v)$ being the shortest-path distance between u and v in G with edge weights given by c. By construction we have $\mathsf{OPT}_{\vec{G}'} \leq \mathsf{OPT}_{\vec{G}}$. We find a closed tour C in $\vec{G}'$ with $\mathsf{cost}_{\vec{G}'}(C) = \mathsf{OPT}_{\vec{G}'}$ in time $2^{O(|V(\vec{G}')|)} \cdot n^{O(1)} = 2^{O(a+g)} \cdot n^{O(1)}$. By composing C with the k' closed walks $C_1, \ldots, C_{k'}$, and shortcutting as in [11], we obtain a solution for the original instance, of total cost $O(a \cdot g + p^2) \cdot \mathsf{OPT}_{\vec{G}}$. ◀

We are now ready to prove the main algorithmic result of this paper.

Proof of Theorem 1. We may assume $k \geq 2$ since otherwise the assertion follows by Theorem 5. We may also assume w.l.o.g. that $p \geq 2$. Let $\vec{G}$ be a (a, g, k, p)-nearly embeddable digraph. It suffices to show that there exists a polynomial time computable $(a, g + k - 1, 1, 2p)$-nearly embeddable digraph $\vec{G}'$ with $V(\vec{G}') = V(\vec{G})$ such that for all $u, v \in V(\vec{G})$ we have $d_{\vec{G}}(u, v) = d_{\vec{G}'}(u, v)$. We compute $\vec{G}'$ as follows. Let $\vec{H}_1, \ldots, \vec{H}_k$ be the vortices of $\vec{G}$ and let $\vec{F}_1, \ldots, \vec{F}_k$ be the faces on which they are attached. For each $i \in \{1, \ldots, k\}$ pick distinct $e_i, f_i \in E(\vec{F}_i)$, with $e_i = \{w_i, w_i'\}$, $f_i = \{z_i, z_i'\}$. There exists a path decomposition $B_{i,1}, \ldots, B_{i,\ell_i}$ of $\vec{H}_i$, of width at most $2p$, and such that $B_{i,1} = e_i$, and $B_{i,\ell_i} = f_i$. For each $i \in \{1, \ldots, k - 1\}$, we add edges (w_{i+1}, z_i), (z_i, w_{i+1}), (w_{i+1}', z_i'), and (z_i', w_{i+1}') to $\vec{G}'$, and we set their length to be equal to the shortest path distance between their endpoints in $\vec{G}$. We also add a handle connecting punctures in the disks bounded by $\vec{F}_i$ and $\vec{F}_{i+1}$ respectively, and we route the four new edges along this handle. Since we add $k - 1$ handles in total the Euler genus of the underlying surface increases by at most $k - 1$. We let $\vec{H}$ be the single vortex in $\vec{G}'$ with $V(\vec{H}) = \bigcup_{i=1}^{k} V(\vec{H}_i)$ and $E(\vec{H}) = \left(\bigcup_{i=1}^{k} E(\vec{H}_i)\right) \cup \left(\bigcup_{i=1}^{k-1} \{(w_{i+1}, z_i), (z_i, w_{i+1}), (w_{i+1}', z_i'), (z_i', w_{i+1}')\}\right)$. It is immediate that

$$B_{1,1}, \ldots, B_{1,\ell_1}, \{f_1, e_2\}, B_{2,1}, \ldots, B_{2,\ell_2}, \{f_2, e_3\}, \ldots, B_{k,1}, \ldots, B_{k,\ell_k}$$

is a path decomposition of $\vec{H}$ of width at most $2p$. Thus $\vec{G}'$ is $(a, g + k - 1, 1, 2p)$-nearly embeddable, which concludes the proof. ◀

5 Combining the Held-Karp LP with the dynamic program

In this Section we show how to combine the dynamic program that finds an optimal closed walk traversing all the vertices in a vortex, with the Held-Karp LP. The following summarizes our exact algorithm for traversing the vortex in a nearly-embeddable graph. The proof of Theorem 6 is deferred to Section 14.

▶ **Theorem 6.** *Let $\vec{G}$ be an n-vertex $(a, g, 1, p)$-nearly embeddable graph and let $\vec{H}$ be the single vortex of $\vec{G}$. Then there exists an algorithm which computes a walk $\vec{W}$ visiting all vertices in $V(\vec{H})$ of total length at most $\mathsf{OPT}_{\vec{G}}(V(\vec{H}))$ in time $n^{O((a+p)g^4)}$.*

▶ **Definition 7** ($\vec{W}$-augmentation)**.** Let $\vec{G}$ be a directed graph. Let $x : E(\vec{G}) \to \mathbb{R}$ and let $\vec{W} \subseteq \vec{G}$. We define the *$\vec{W}$-augmentation* of x to be the function $x' : E(\vec{G}) \to \mathbb{R}$ such that for all $e \in E(\vec{G})$ we have

$$x'(e) = \begin{cases} x(e)+1 & \text{if } e \in E(\vec{W}) \\ x(e) & \text{otherwise} \end{cases}$$

The following summarizes the main technical result for computing a thin spanning subgraph in a nearly embeddable graph. The proof of Lemma 8 is deferred to Section 9.

▶ **Lemma 8.** *Let $\vec{G}$ be a $(a, g, 1, p)$-nearly embeddable digraph, let $\vec{H}$ be its vortex, and let $\vec{W}$ be a walk in $\vec{G}$ visiting all vertices in $V(\vec{H})$. Let G, H, and W be the symmetrizations of $\vec{G}$, $\vec{H}$, and $\vec{W}$ respectively. Let $z : E(G) \to \mathbb{R}_{\geq 0}$ be α-thick for some $\alpha \geq 2$, and $\vec{W}$-dense. Then there exists a polynomial time algorithm which given $\vec{G}$, $\vec{H}$, A, $\vec{W}$, z, and an embedding of $\vec{G} \setminus (A \cup \vec{H})$ into a surface of genus g, outputs a subgraph $S \subseteq G \setminus H$, satisfying the following conditions:*

1. *$W \cup S$ is a spanning subgraph of G and has $O(a+g)$ connected components.*
2. *$W \cup S$ is $O(a \cdot g + p^2)$-thin w.r.t. z.*

We are now ready to prove the main result of this section.

Proof of Lemma 4. Let $\vec{H}$ be the single vortex of $\vec{G}$. We compute an optimal solution $y : E(\vec{G}) \to \mathbb{R}$ for the Held-Karp LP for $\vec{G}$. We find a tour $\vec{W}$ in $\vec{G}$ visiting all vertices in $V(\vec{H})$, with $\mathsf{cost}_{\vec{G}}(\vec{W}) = O(\mathsf{OPT}_{\vec{G}})$ using Theorem 6. Let $x : E(\vec{G}) \to \mathbb{R}$ be the $\vec{W}$-augmentation of y. Since for all $e \in E(\vec{G})$ we have $x(e) \geq y(e)$, it follows that x is a feasible solution for the Held-Karp LP. Moreover since $\mathsf{cost}_{\vec{G}}(\vec{W}) = O(\mathsf{OPT}_{\vec{G}})$, we obtain that $\mathsf{cost}_{\vec{G}}(x) = \mathsf{cost}_{\vec{G}}(y) + \mathsf{cost}_{G}(\vec{W}) = O(\mathsf{OPT}_{\vec{G}})$. Let z be the symmetrization of x.

Note that z is 2-thick and $\vec{W}$-dense. Therefore, by Lemma 8 we can find a subgraph $S \subseteq G \setminus H$ such that $T = W \cup S$ is a $O(a \cdot g + p^2)$-thin spanning subgraph of G (w.r.t. z), with at most $O(g + a)$ connected components. Therefore, there exists a constant α such that for every $U \subseteq V(G)$ we have $|T \cap \delta(U)| \leq \alpha \cdot (a \cdot g + p^2) \cdot z(\delta(U))$. We can assume that $\alpha \geq 1$. Now we follow a similar approach to [8].

Let $m = \lfloor n^2/\alpha \rfloor$. We define a sequence of functions $z_0, \ldots, z_m$, and a sequence of spanning forests $T_1, \ldots, T_m$ satisfying the following conditions.

1. For any $i \in \{0, \ldots, m\}$, z_i is non-negative, 2-thick and $\vec{W}$-dense.
2. For any $i \in \{1, \ldots, m\}$, T_i has at most $O(a+g)$ connected components.
3. For every $U \subseteq V(G)$ we have $|T_{i+1} \cap \delta(U)| \leq \alpha \cdot (a \cdot g + p^2) \cdot z_i(\delta(U))$.

We set $z_0 = 3\lfloor zn^2 \rfloor / n^2$. Now suppose for $i \in \{0, \ldots, m-1\}$ we have defined z_i. We define z_{i+1} and T_{i+1} as follows. We apply Lemma 8 and we obtain a subgraph T_{i+1} of G with at most $O(a+g)$ connected components such that for every $U \subseteq V(G)$ we have $|T_{i+1} \cap \delta(U)| \leq \alpha \cdot (a \cdot g + p^2) \cdot z_i(\delta(U))$. Also, for every $e \in E(G)$ we set $z_{i+1}(e) = z_i(e)$ if $e \notin T_{i+1}$, and $z_{i+1}(e) = z_i(e) - 1/n^2$ if $e \in T_{i+1}$. Now by using the same argument as in [8], we obtain that z_{i+1} is non-negative and 2-thick. By the construction, we know that z is $\vec{W}$-dense and thus for all $(u, v) \in \vec{W}$ we have $z_0(\{u, v\}) \geq 3$. Note that for all $e \in E(G)$ we have $z_{i+1}(e) \geq z_i(e) - 1/n^2$. Thus for all $e \in E(G)$ and for all $i \in \{0, \ldots, m\}$ we have $z_i(e) \geq 2$. Therefore for all $i \in \{0, \ldots, m\}$ we have that z_i is $\vec{W}$-dense.

Now, similar to [8] we set the desired S to be the subgraph T_i that minimizes $\mathsf{cost}_G(T_i)$, which implies that S is a $(O(a \cdot g + p^2), O(1))$-thin spanning subgraph with at most $O(a+g)$ connected components. ◀

6 Thin trees in 1-apex graphs

In this section we show how to compute thin trees in 1-apex graphs. The following is implicit in the work of Oveis Gharan and Saberi [13].

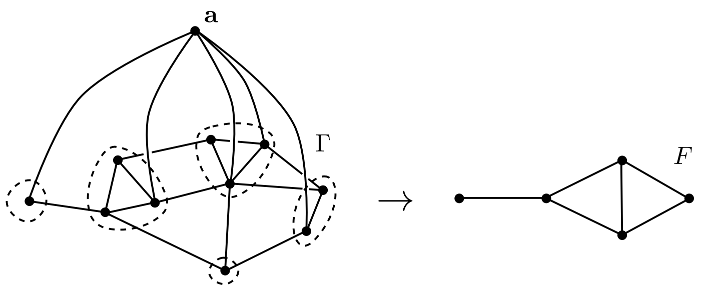
▶ **Theorem 9.** *If G is a planar graph and z is an α-thick weight function on the edges of G for some $\alpha > 0$, then there exists a $10/\alpha$-thin spanning tree in G w.r.t. z.*

For the remainder of this section, let G be an 1-apex graph with planar part Γ and apex $\mathbf{a}$. Let z be a 2-thick weight function on the edges of G. We will find a $O(1)$-thin spanning tree in G (w.r.t. z). We describe an algorithm for finding such a tree in polynomial time. The algorithm proceeds in five phases.

Phase 1. We say that a cut U is *tiny* (w.r.t. z) if $z(\delta(U)) < 0.1$. We start with Γ and we proceed to partition it via tiny cuts. Each time we find a tiny cut U, we partition the remaining graph by deleting all edges crossing U. This process will stop in at most n steps. Let Γ' be the resulting subgraph of Γ where $V(\Gamma') = V(\Gamma)$ and $E(\Gamma') \subset E(\Gamma)$.

Phase 2. By the construction, we know that there is no tiny cuts in each connected component of Γ'. Therefore, following [13], in each connected component C of Γ', we can find a $O(1)$-thin spanning tree T_C (w.r.t. z). More specifically, we will find a 100-thin spanning tree in each of them.

Phase 3. We define a graph F with $V(F)$ being the set of connected components of Γ' and $\{C, C'\} \in E(F)$ iff there exists an edge between some vertex in C and some vertex in C' in Γ. We set the weight of $\{C, C'\}$ to be $z(C, C')$. We call F the *graph of components*.



We define a graph G' obtained from G by contracting every connected component of Γ' into a single vertex. We remark that we may get parallel edges in G'.

Phase 4. In this phase, we construct a tree T' in G'. We say that a vertex in F is *originally heavy*, if it has degree of at most 15 in F. Since F is planar, the minimum degree of F is at most 5. We contract all vertices in $V(G') \setminus \{\mathbf{a}\}$ into the apex sequentially. In each step, we find a vertex in F with degree at most 5, we contract it to the apex in G', and we delete it from F. Since the remaining graph F is always planar, there is always a vertex of degree at most 5 in it, and thus we can continue this process until all vertices of $V(G') \setminus \{\mathbf{a}\}$ are contracted into the apex.

Initially we consider all vertices of F having no parent. In each step when we contract a vertex C with degree at most 5 in F to the apex in G', for each neighbor C' of C in F, we make C the *parent* of C' if C' does not have any parents so far. Note that each vertex in F can be the parent of at most 5 other vertices, and can have at most one parent.

Every time we contract a vertex C to the apex, we add an edge e to T'. If C is originally heavy, we add an arbitrary edge e from C to the apex; we will show in Lemma 11 that $z(C, \{a\}) \geq 0.5$, which implies that such an edge always exists in G. Otherwise, we add an arbitrary edge from C to its parent (which is a neighbor vertex, therefore such an edge exists in G). We will show in the next section that each vertex in F is originally heavy or it has a parent (or both). Therefore, T' is a tree on G'.

Phase 5. In this last phase we compute a tree T in G. We set $E(T) = E(T') \cup \bigcup_{C \in V(F)} E(T_C)$. We prove in the next subsection that T is a $O(1)$-thin spanning tree in G.

6.1 Analysis

We next show that T is a $O(1)$-thin spanning tree in G.

▶ **Lemma 10.** *The weight of every edge in F is less than* 0.1.

Proof. Let $\{C, C'\} \in E(F)$. By construction, each component of Γ' is formed by finding a tiny cut in some other component. Suppose C was formed either simultaneously with C' or later than C' by finding a tiny cut in some C''. If $C' \subseteq C''$ then $z(C, C') < z(\delta(C)) < 0.1$. Otherwise, the total weight of edges from C to C' is a part of a tiny cut which means that $z(C, C') < 0.1$. ◀

▶ **Lemma 11.** *Let C be an originally heavy vertex in F. Then $z(C, \{\mathbf{a}\}) \geq 0.5$.*

Proof. For every neighbor C' of C in F, by Lemma 10 we have that the weight of $\{C, C'\}$ is less than 0.1. By the assumption on z, we have that $z(\delta(C)) \geq 2$. Now since C has degree of at most 15, we have that $z(C, \{\mathbf{a}\}) \geq 0.5$, as desired. ◀

▶ **Lemma 12.** *Each vertex $C \in V(F)$ is originally heavy or it has a parent (both cases might happen for some vertices).*

Proof. Let $C \in V(F)$. If it is originally heavy, we are done. Otherwise, it has degree of at least 15. We know that all vertices in F are going to be contracted to $\mathbf{a}$ at some point, and we only contract vertices with degree at most 5 in each iteration. This means that at least 10 other neighbors of C were contracted to the apex before we decided to contract C to the apex. Therefore one of them is the parent of C. ◀

▶ **Lemma 13.** *T is a spanning tree in G.*

Proof. Suppose G has n vertices and F has m vertices. After the second phase of our algorithm, we obtain a spanning forest on Γ with m components and $n - m - 1$ edges. Each time we contract a vertex of F to the apex, we add a single edge to T. Therefore, T has $n - 1$ edges. It is now sufficient to show that T is connected.

We will show that for every vertex u in Γ, there is path between u and $\mathbf{a}$ in T. Let u be a vertex of Γ. Suppose u is in some component C_u which is a vertex of F. If C_u is originally heavy, then there is an edge e in T between a vertex $v \in C_u$ and the apex. Since we have a spanning tree in C_u, there is a path between u and v in T. Therefore, there is path between u and the apex a in T.

Otherwise, C_u must have some parent C_{u_1} and there is an edge between these two components. Therefore, there is a path between u and each vertex of these two components. Now, the same argument applies for C_{u_1}. Either it is originally heavy or it has a parent C_{u_2}.

If it is originally heavy, we are done. Otherwise, we use the same argument for C_{u_2}. Note that by construction and the definition of a parent, we do not reach the same component in this sequence. Therefore, at some point, we reach a component C_{u_k} which is originally heavy and we are done. ◀

Now we are ready to show that T is a $O(1)$-thin tree in G (w.r.t. z). We have to show that there exists some constant α such that for every cut U, $|E(T) \cap \delta(U)| \leq \alpha \cdot z(\delta(U))$. Let U be a cut in G. We can assume w.l.o.g. that $\mathbf{a} \notin U$, since otherwise we can consider the cut $V(G) \setminus U$. We partition $E(T) \cap \delta(U)$ into three subsets:
1. $T_1 = \{\{\mathbf{a}, v\} \in E(T) \cap \delta(U) : v \in V(\Gamma)\}$.
2. $T_2 = \{\{u, v\} \in E(T) \cap \delta(U) : u$ and v are in the same component of $\Gamma'\}$.
3. $T_3 = \{\{u, v\} \in E(T) \cap \delta(U) : u$ and v are in different components of $\Gamma'\}$.

▶ **Lemma 14.** *There exists some constant α_1 such that $|T_1| \leq \alpha_1 \cdot z(\delta(U))$.*

Proof. Let $e = \{\mathbf{a}, v\} \in T_1$ where $v \in V(\Gamma)$. Let $C_v \in V(F)$ such that $v \in C_v$. By the construction of T, C_v is originally heavy. If $C_v \subseteq U$, we can charge e to $z(C_v, \{\mathbf{a}\})$, which we know is at least 0.5. Otherwise suppose C_v is not a subset of U. By the assumption we have $\mathbf{a} \notin U$ and thus $v \in U$ which implies that $U \cap C_v \neq \emptyset$. By the construction, we know that there is no tiny cuts in C_v. Therefore, $z(\delta(U) \cap E(G[C_v])) \geq 0.1$. Thus we can charge e to the total weight of the edges in $\delta(U) \cap E(G[C_v])$. Note that for each $C_v \in V(F)$, there is at most one edge in T_1 between $\mathbf{a}$ and C_v. Therefore we have that $|T_1| \leq 10 \cdot z(\delta(U))$. ◀

▶ **Lemma 15.** *There exists some constant α_2 such that $|T_2| \leq \alpha_2 \cdot z(\delta(U))$.*

Proof. We have

$$\begin{aligned} |T_2| &= \left| \bigcup_{C \in V(F)} (E(C) \cap T_2) \right| = \left| \bigcup_{C \in V(F)} (E(C) \cap E(T) \cap \delta(U)) \right| \\ &= \left| \bigcup_{C \in V(F)} (E(T_C) \cap \delta(U)) \right| \leq \sum_{C \in V(F)} 100 \cdot z(\delta(U) \cap E(C)) \leq 100 \cdot z(\delta(U)), \end{aligned}$$

concluding the proof. ◀

▶ **Lemma 16.** *There exists some constant α_3 such that $|T_3| \leq \alpha_3 \cdot z(\delta(U))$.*

Proof. We partition T_3 into three subsets:
1. $T_{31} = \{\{u, v\} \in T_3 : u \in U, v \notin U, \exists C_u, C_v \in V(F)$ s.t. $u \in C_u, v \in C_v, U \cap C_v \neq \emptyset\}$.
2. $T_{32} = \{\{u, v\} \in T_3 : u \in U, v \notin U, \exists C_u, C_v \in V(F)$ s.t. $u \in C_u, v \in C_v, U \cap C_u \neq \emptyset\}$.
3. $T_{33} = \{\{u, v\} \in T_3 : u \in U, v \notin U, \exists C_u, C_v \in V(F)$ s.t. $u \in C_u, v \in C_v, U \cap C_v = \emptyset, C_u \subseteq U\}$.

First for each $e = \{u, v\} \in T_{31}$ where $v \in C_v$ for some $C_v \in V(F)$, we have that $U_v = U \cap C_v$ is a cut in C_v which is not tiny. By the construction, C_v can be the parent of at most five other vertices in F and it can have at most one parent. Therefore, there are at most six edges in T_3 with a vertex in C_v. So we can charge e and at most five other edges to $z(\delta(U_v))$. Since $z(\delta(U_v)) \geq 0.1$ we get $|T_{31}| \leq 60 \cdot z(\delta(U))$.

Second for each $e = \{u, v\} \in T_{32}$ where $u \in C_u$ for some $C_u \in V(F)$, we have that $U_u = U \cap C_u$ is a cut in C_u which is not tiny. Therefore, the same argument for C_v as in the first case, applies here for C_u and we get $|T_{32}| \leq 60 \cdot z(\delta(U))$.

Finally, for T_{33} we need to find a constant α_{33} such that $|T_{33}| \leq \alpha_{33} \cdot z(\delta(U))$. First, we define a new cut U_1 as follows. For every $C \in V(F)$ with $C \cap U \neq \emptyset$, if $C \cap U \neq C$, we add all the other vertices of C to U and we say that C is *important*. This process leads to a new cut U_1 such that for every $C \in F$, either $C \cap U_1 = \emptyset$ or $C \subseteq U_1$. Let $U_2 = \{C \in V(F) : C \subseteq U_1\}$. Let $X = \{C \in V(F) : C \notin U_2\}$ and $Y = X \cup \{\mathbf{a}\}$.

Let

$$T_{331} = \{\{u, v\} \in T_{33} : u \in U, u \in C \text{ for some } C \in V(F) \text{ with } \deg_{F[U_2]}(C) < 19\}.$$

Let also $T_{332} = T_{33} \setminus T_{331}$.

For each edge $e = \{u, v\} \in T_{331}$ where $u \in U$ and $u \in C_u$ for some $C_u \in V(F)$, we have $\deg_{F[U_2]}(C_u) < 19$. By Lemma 10, we know that for any $C, C' \in V(F)$, $z(C, C') \leq 0.1$. Therefore, we get $z(C_u, Y) \geq 0.2$. Note that there are at most six edges in T_{331} with a vertex in C_u. So we can charge e and at most five other edges to $z(C_u, Y)$. Therefore, $|T_{331}| \leq 30 \cdot z(\delta(U))$.

Let $V_1 = \{C \in U_2 : \deg_{F[U_2]}(C) \geq 19\}$ and $V_2 = \{C \in U_2 : \deg_{F[U_2]}(C) \leq 5\}$. By Euler's formula, we know that the average degree of a planar graph is at most 6. Since $F[U_2]$ is planar, we get $|V_1| \leq |V_2|$. For any $C \in V_2$, if C is important, then $C \cap U$ is a cut for C and we have $z(C \cap U, Y) \geq 0.1$. If C is not important, then we have $z(C, Y) \geq 1.5$. Note that for any $C' \in V_1$, there are at most six edges in T_{332} with a vertex in C'. Therefore, we have $|T_{332}| \leq 60 \cdot z(\delta(U))$.

Now since $T_3 = T_{31} \cup T_{32} \cup T_{331} \cup T_{332}$, we have $|T_3| \leq 210 \cdot z(\delta(U))$ completing the proof. ◀

▶ **Lemma 17.** *T is a $O(1)$-thin spanning tree in G.*

Proof. By Lemma 13 we know that T is a spanning tree. For any $U \subseteq V(G)$, by Lemmas 14, 15 and 16 we get $|T| \leq 320 \cdot z(\delta(U))$. This completes the proof. ◀

We are now ready to prove the main result of this Section.

▶ **Theorem 18.** *Let G be a 1-apex graph and let $z : E(G) \to \mathbb{R}_{\geq 0}$ be β-thick for some $\beta > 0$. Then there exists a polynomial time algorithm which given G and z outputs a $O(1/\beta)$-thin spanning tree in G (w.r.t. z).*

Proof. For $\beta \geq 2$, by Lemma 17 we know that we can find a 320-thin spanning tree in G. For any β with $0 < \beta < 2$, the assertion follows by scaling z by a factor of $2/\beta$. ◀

7 Thin forests in graphs with many apices

Let $a \geq 1$. In this section, we describe an algorithm for finding thin-forests in an a-apex graph. The high level approach is analogous to the case of 1-apex graphs. We construct a similar graph F of components and contract each vertex of F to some apex.

Let $e_0 = \{u_0, v_0\} \in E(G)$. Let G' be obtained from G by contracting e_0. We define a new weight function z' on the edges of G' as follows. For any $\{u, v_0\} \in E(G)$, we set $z'(\{u, v_0\}) = z(\{u, v_0\}) + z(\{u, u_0\})$. For any other edge e we set $z'(e) = z(e)$. We say that z' is *induced* by z. Similarly, when G' is obtained by contracting a subset X of edges in G, we define z' by inductively contracting the edges in X in some arbitrary order.

For the remainder of this section let G be a a-apex graph with the set of apices $A = \{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_a\}$. Let Γ be the planar part of G. Let z be a 2-thick weight function on the edges of G. The algorithm proceeds in 5 phases.

Phase 1. We say that a cut U is *tiny* (w.r.t. z) if $z(\delta(U)) < 1/(100 \cdot a)$. Similar to the case of 1-apex graphs, we start with Γ and repeatedly partition it via tiny cuts until there are no more such cuts and we let Γ' be the resulting graph.

Phase 2. For each connected component C of Γ' we find a $O(a)$-thin tree T_C using Theorem 9.

Phase 3. We define F and G' exactly the same way as in the case of 1-apex graphs.

▶ **Lemma 19.** *For every $\{C, C'\} \in E(F)$, we have $z(C, C') \leq 1/(100 \cdot a)$.*

Proof. The same argument as in Lemma 10 applies here. The only difference here is that a cut U is tiny if $z(\delta(U)) < 1/(100 \cdot a)$. ◀

Phase 4. We construct a forest T' on G'. Let $m = |V(F)|$. We define a sequence of planar graphs $F_0, F_1, \cdots, F_m$, a sequence of graphs $G'_0, G'_1, \cdots, G'_m$ and a sequence of weight functions $z_0, z_1, \cdots, z_m$ as follows. Let $F_0 = F$, $G'_0 = G'$ and $z_0 = z$. We also define a sequence of forests $P_0, \ldots, P_m$ where each P_j contains a tree rooted at each $\mathbf{a}_i \in A$. We set P_0 to be the forest that contains a tree for each $\mathbf{a}_i \in A$ and with no other vertices.

Let $C \in V(F_j)$ for some j. For any $\mathbf{a}_i \in A$, we say that C is *$\mathbf{a}_i$-heavy* in F_j if $z_j(C, \{\mathbf{a}_i\}) \geq 1/a$. Let $C' \in V(F)$. For any $\mathbf{a}_i \in A$, we say that C' is *originally $\mathbf{a}_i$-heavy* if C' is $\mathbf{a}_i$-heavy in F.

We maintain the following inductive invariant:

(I1) For any $j \in \{0, \ldots, m-1\}$, let $v \in V(F_j)$ be a vertex of minimum degree. Then either there exists some $\mathbf{a}_i \in A$ such that v is originally $\mathbf{a}_i$-heavy or $v \in V(P_j)$.

Consider some $i \in \{0, \ldots, m-1\}$. Let $v_i \in V(F_i)$ be a vertex with minimum degree. If v_i is originally $\mathbf{a}_j$-heavy for some $\mathbf{a}_j \in A$, then we contract v_i to $\mathbf{a}_j$. Otherwise, by the inductive invariant (I1), we have that $v_i \in V(P_i)$. Thus there exists a tree in P_i containing v_i that is rooted in some $\mathbf{a}_j \in A$; we contract v_i to $\mathbf{a}_j$. In either case, by contracting v_i to $\mathbf{a}_j$ we obtained G'_{i+1} from G'_i. We also delete v_i from F_i to obtain F_{i+1}. We let z_{i+1} be the weight function on G'_{i+1} induced by z_i.

Finally, we need to define P_{i+1}. If v_i was originally $\mathbf{a}_j$-heavy then we add v_i to P_i via an edge $\{v_i, \mathbf{a}_j\}$. For each $u \in V(F_i)$ that is a neighbor of v_i, and is not in $V(P_i)$, we add u to P_{i+1} by adding the edge $\{v_i, u\}$ iff the following conditions hold:

(i) For all $\mathbf{a}_r \in A$, we have that u is not $\mathbf{a}_r$-heavy in F_i.

(ii) u is $\mathbf{a}_j$-heavy in F_{i+1}.

In this case we say that v is the *parent* of u. This completes the description of the process that contracts each vertex in $V(G')$ into some apex.

▶ **Lemma 20.** *Let $j \in \{0, 1, \ldots, m-1\}$. Let $C \in V(F_j)$ be a vertex with minimum degree. Then there exists $\mathbf{a}_i \in A$ such that C is $\mathbf{a}_i$-heavy in F_j.*

Proof. Since C has at most 5 neighbors in F_j, by Lemma 19 we have $z_j(C, A) \geq 2 - 5/(100 \cdot a) \geq 1$. Therefore by averaging, there exists an apex $\mathbf{a}_i$ such that $z_j(C, \{\mathbf{a}_i\}) \geq 1/a$. ◀

▶ **Lemma 21.** *For any $j \in \{0, \ldots, m\}$ and for any $v \in V(\Gamma) \cap V(P_j)$, we have* $\deg_{P_j}(v) \leq 6$.

Proof. By the construction, in each step we pick a vertex of minimum degree and contract it into some apex. Since F_i is planar, its minimum degree is at most 5. This means that for any $v \in V(\Gamma) \cap V(P_j)$, v can be the parent of at most five other vertices and can have at most one parent. This completes the proof. ◀

▶ **Lemma 22.** *The inductive invariant (I1) is maintained.*

Proof. For any $j \in \{0, \dots, m-1\}$, let $v \in V(F_j)$ be a vertex of minimum degree. If there exists some $\mathbf{a}_i \in A$ such that v is originally $\mathbf{a}_i$-heavy, then we are done. Suppose for all $\mathbf{a}_i \in A$, v is not originally $\mathbf{a}_i$-heavy. By Lemma 20 we know that there exists some $\mathbf{a}_l \in A$ such that v is $\mathbf{a}_l$-heavy in F_j. Let $j^* \in \{1, \dots, j\}$ be minimum such that v is not $\mathbf{a}_t$-heavy in F_{j^*-1} for all $\mathbf{a}_t \in A$, and v is $\mathbf{a}_{t'}$-heavy in F_{j^*} for some $\mathbf{a}_{t'} \in A$. Let $u \in V(F_{j^*-1})$ be vertex that is contracted to some apex in step j^*. It follows by construction that u is the parent of v in P_{j^*}. Since $j \geq j^*$ it follows that $v \in V(P_j)$, concluding the proof. ◀

Now we are ready to describe how to construct T' in G'. For any $l \in \{0, \dots, m-1\}$, let $C \in V(F_l)$ be a vertex of minimum degree. If C is originally $\mathbf{a}_i$-heavy for some $\mathbf{a}_i \in A$ and we contract C to $\mathbf{a}_i$, we pick an arbitrary edge e between C and $\mathbf{a}_i$ and we add it to T'. Otherwise, by Lemma 22 we have $C \in V(P_l)$. This means that C has a parent C'. In this case, we pick an arbitrary edge e between C and C' and we add it to T'.

Phase 5. We construct a forest T in G the same way as in the 1-apex case. We set $E(T) = E(T') \cup \bigcup_{C \in V(F)} E(T_C)$.

This completes the description of the algorithm.

7.1 Analysis

By the construction, T has a connected components. We will show that T is a $O(a)$-thin spanning forest. Let U be a cut in G. Similar to the 1-apex case, we partition $E(T) \cap \delta(U)$ into three subsets:

1. $T_1 = \{\{\mathbf{a}_i, v\} \in E(T) \cap \delta(U) : \mathbf{a}_i \in A, v \in V(\Gamma)\}$.
2. $T_2 = \{\{u, v\} \in E(T) \cap \delta(U) : u \text{ and } v \text{ are in the same component of } \Gamma'\}$.
3. $T_3 = \{\{u, v\} \in E(T) \cap \delta(U) : u \text{ and } v \text{ are in different components of } \Gamma'\}$.

▶ **Lemma 23.** *There exists a constant α_1 such that $|T_1| \leq \alpha_1 \cdot a \cdot z(\delta(U))$.*

Proof. A similar argument as in the case of 1-apex graph applies here with two differences. First, a cut U is tiny if $z(\delta(U)) < 1/(100 \cdot a)$. Second, for any $\mathbf{a}_i \in A$ and $C \in V(F)$ where C is originally $\mathbf{a}_i$-heavy, we have that $z(C, \{\mathbf{a}_i\}) \geq 1/a$. Therefore, we get $|T_1| \leq 100 \cdot a \cdot z(\delta(U))$. ◀

▶ **Lemma 24.** *There exists a constant α_2 such that $|T_2| \leq \alpha_2 \cdot a \cdot z(\delta(U))$.*

Proof. Again, a similar argument as in the case of 1-apex graphs applies here. The only difference here is the definition of tiny cut. Therefore, we get $|T_2| \leq 100 \cdot a \cdot z(\delta(U))$. ◀

▶ **Lemma 25.** *There exists a constant α_3 such that $|T_3| \leq \alpha_3 \cdot a \cdot z(\delta(U))$.*

Proof. Similar to the 1-apex case, we partition T_3 into three subsets:

1. $T_{31} = \{\{u, v\} \in T_3 : u \in U, v \notin U, \exists C_u, C_v \in V(F) \text{ s.t. } u \in C_u, v \in C_v, U \cap C_v \neq \emptyset\}$.
2. $T_{32} = \{\{u, v\} \in T_3 : u \in U, v \notin U, \exists C_u, C_v \in V(F) \text{ s.t. } u \in C_u, v \in C_v, U \cap C_u \neq \emptyset\}$.
3. $T_{33} = \{\{u, v\} \in T_3 : u \in U, v \notin U, \exists C_u, C_v \in V(F) \text{ s.t. } u \in C_u, v \in C_v, U \cap C_v = \emptyset, C_u \subseteq U\}$.

The arguments for T_{31} and T_{32} are the same as in 1-apex graphs. The only difference here is that a cut U is tiny if $z(\delta(U)) < 1/(100 \cdot a)$. Therefore, we have $|T_{31}| \leq 600 \cdot a \cdot z(\delta(U))$ and $|T_{32}| \leq 600 \cdot a \cdot z(\delta(U))$.

Now for T_{33} we want to find a constant α_{33} such that $|T_{33}| \leq \alpha_{33} \cdot a \cdot z(\delta(U))$. We define two new cuts U_1 and U_2 as follows. For every $C \in V(F)$ with $C \cap U \neq \emptyset$, if $C \cap U \neq C$, we add all other vertices of C to U (delete all other vertices of C from U) to obtain U_1 (U_2) and we say that C is *U-important*. Let $U_1' = \{C \in V(F) : C \subseteq U_1\}$ and $U_2' = \{C \in V(F) : C \subseteq U_2\}$.

For any $e = \{u, v\} \in T_{33}$ where $u \in U$, $v \notin U$, $u \in C_u$ and $v \in C_v$ for some $C_u, C_v \in V(F)$, by the construction of T_3, we have that both C_u and C_v have been contracted to the same apex $\mathbf{a}_i$ for some $\mathbf{a}_i \in A$. Let $j \in \{0, \dots, m\}$ be the step during which C_u is contracted to $\mathbf{a}_i$. Let $B = \{C \in V(F) : C \text{ is contracted to } \mathbf{a}_i\}$. Let D be the connected component of $F[B]$ containing C_u. Let $D^{\mathsf{in}}_{U_1'} = D[U_1']$, $D^{\mathsf{out}}_{U_1'} = D[V(D) \setminus U_1']$, $D^{\mathsf{in}}_{U_2'} = D[U_2']$, $D^{\mathsf{out}}_{U_2'} = D[V(D) \setminus U_2']$.

We consider the following two cases:

Case 1: $\mathbf{a}_i \notin U$. We know that $C_u \in U$. By the construction, we have that $z_j(C_u, \{\mathbf{a}_i\}) \geq 1/a$. If $z_0(C_u, \{\mathbf{a}_i\}) \geq 1/(100 \cdot a)$, we can charge e to $z_0(C_u, \{\mathbf{a}_i\})$ and we know that there are at most six edges in T_{33} with a vertex in C_u.
Otherwise, we have that $z_0(C_u, (V(D) \setminus C_u)) \geq 99/(100 \cdot a)$. If $z_0(C_u, (V(D^{\mathsf{out}}_{U_1'}) \setminus C_u)) \geq 1/(100 \cdot a)$, then by the construction of $D^{\mathsf{out}}_{U_1'}$ we get $z_0(C_u, (G \setminus U)) \geq 1/(100 \cdot a)$. Therefore we can charge e to $z_0(C_u, (G \setminus U))$.
Otherwise, we have that $z_0(C_u, (V(D^{\mathsf{in}}_{U_1'}) \setminus C_u)) \geq 98/(100 \cdot a)$. This implies that $\deg_{D^{\mathsf{in}}_{U_1'}}(C_u) \geq 98$. Now note that $D^{\mathsf{in}}_{U_1'}$ is a planar graph and its average degree is at most 6. Now a similar argument as in the 1-apex case applies here. Let $V_1 = \{C \in D^{\mathsf{in}}_{U_1'} : \deg_{D^{\mathsf{in}}_{U_1'}}(C) \geq 98\}$. Let $V_2 = \{C \in D^{\mathsf{in}}_{U_1'} : \deg_{D^{in}_{U_1'}}(C) \leq 5\}$. By planarity of $D^{\mathsf{in}}_{U_1'}$, we have that $|V_1| \leq |V_2|$. For any $C \in V_2$, if C is U-important, then $C \cap U$ is a cut for C which is not tiny. Therefore we have $z(C \cap U, G \setminus U) \geq 1/(100 \cdot a)$. If C is not U-important, we have $z(C, G \setminus U) \geq 95/(100 \cdot a)$. Now note that for any $C' \in V_1$ there are at most six edges in T_{33} with a vertex in C'. Therefore we have $|T_{33}| \leq 1000 \cdot a \cdot z(\delta(U))$.

Case 2: $\mathbf{a}_i \in U$. Let $X_1 = \{C \in D^{out}_{U_2'} : \deg_{D^{out}_{U_2'}}(C) \geq 98\}$. Let $X_2 = \{C \in D^{out}_{U_2'} : \deg_{D^{out}_{U_2'}}(C) \leq 5\}$. We know that $C_v \notin U$. We follow a similar approach as in the first case by considering U_2', $D^{\mathsf{in}}_{U_2'}$ and $D^{\mathsf{out}}_{U_2'}$. The same argument applies here by replacing U_1', $D^{\mathsf{in}}_{U_1'}$, $D^{\mathsf{out}}_{U_1'}$, X_1 and X_2 with U_2', $D^{\mathsf{out}}_{U_2'}$, $D^{\mathsf{in}}_{U_2'}$, V_1 and V_2 respectively. Therefore, we get $|T_{33}| \leq 1000 \cdot a \cdot z(\delta(U))$.

Now from what we have discussed, we have $|T_{31}| \leq 600 \cdot a \cdot z(\delta(U))$, $|T_{32}| \leq 600 \cdot a \cdot z(\delta(U))$ and $|T_{33}| \leq 1000 \cdot a \cdot z(\delta(U))$. Therefore, we get $|T_3| \leq 2200 \cdot a \cdot z(\delta(U))$ completing the proof. ◀

▶ **Lemma 26.** *T is a $O(a)$-thin spanning forest in G with at most a connected components.*

Proof. By the construction, T is a spanning forest and has at most a connected components. Let $\alpha = \alpha_1 + \alpha_2 + \alpha_3$, where α_1, α_2 and α_3 are obtained by Lemmas 23, 24, 25. Therefore for any $U \subset V(G)$, we have $|T| \leq \alpha \cdot a \cdot z(\delta(U)) = 2400 \cdot a \cdot z(\delta(U))$ completing the proof. ◀

We are now ready to prove the main result of this Section.

▶ **Theorem 27.** *Let $a \geq 1$ and let G be a a-apex graph with set of apices $A = \{\boldsymbol{a}_1, \dots, \boldsymbol{a}_a\}$. Let $z : E(G) \to \mathbb{R}_{\geq 0}$ be β-thick for some $\beta > 0$. Then there exists a polynomial time algorithm which given G, A and z outputs a $O(a/\beta)$-thin spanning forest in G (w.r.t. z) with at most a connected components.*

Proof. By Lemma 26, for $\beta \geq 2$, we know that we can find a $(2400a)$-thin spanning forest in G (w.r.t. z) with at most a connected components. For any other $0 < \beta < 2$, the claim follows by scaling z by a factor of $2/\beta$. ◀

8 Thin forests in higher genus graphs with many apices

In this section we generalize our algorithm for computing a thin tree on a graph with many apices, to compute a thin forest in a graph of higher genus and with many apices. The following theorem is implicit in [8].

▶ **Theorem 28** (Erickson and Sidiropoulos [8]). *Let G be a graph with $\mathsf{eg}(G) = g$, and let z be a β-thick weight function on the edges of G for some $\beta \geq 0$. Then there exists a polynomial time algorithm which given G, z, and an embedding of G into a surface of Euler genus g, outputs a $O(1/\beta)$-thin spanning forest in G (w.r.t. z), with at most g connected components.*

In this section, we study the problem in higher genus graphs. First, the following two Lemmas can be obtained by Euler's formula.

▶ **Lemma 29.** *Let G be a graph of genus $g \geq 1$ with $|V(G)| \geq 10g$. Then there exists $v_0 \in V(G)$ with $\deg_G(v_0) \leq 7$.*

▶ **Lemma 30.** *Let G be an n vertex graph of genus $g \geq 1$. Then the average degree of vertices of G is at most $6 + 12(g-1)/n$.*

For the remainder of this section, let G be an a-apex graph with the set of apices $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_a\}$ on a surface of genus g. Let $\Gamma = G \setminus A$, where Γ is a graph of genus g. Let z be a 2-thick weight function on the edges of G. We will find a $O(a \cdot g)$-thin (w.r.t. z) spanning forest in G with at most $O(a+g)$ connected components. The high level approach is similar to the case where Γ was planar. The algorithm proceeds in 5 phases.

Phase 1. We say that a cut U is *tiny* (w.r.t. z) if $z(\delta(U)) < 1/(1000 \cdot a \cdot g)$. We construct Γ' the same way as in Section 7. The only difference here is the definition of tiny cut.

Phase 2. Similar to the planar case, for each connected component C of Γ' we find a $O(a \cdot g)$-thin forest T_C, with at most g connected components, using Theorem 28

Phase 3. We define F and G' the exact same way as in Section 7.

▶ **Lemma 31.** *For every $\{C, C'\} \in E(F)$, we have $z(C, C') \leq 1/(1000 \cdot a \cdot g)$.*

Proof. The same argument as in Lemma 10 applies here. The only difference here is the definition of tiny cut. ◀

Phase 4. We construct a spanning forest T' on G', with at most $a + 10g$ connected components. We follow a similar approach as in the planar case. Let $m = |V(F)| - 10g$. If $m \leq 0$, we set $E(T') = \emptyset$ and we skip to the next phase. Otherwise, we define two sequences of graphs $F_0, F_1, \ldots, F_m$, $G'_0, G'_1, \ldots, G'_m$, a sequence of weight functions $z_0, z_1, \ldots, z_m$, a sequence of forests $P_0, P_1, \ldots, P_m$ satisfying the inductive invariant (I1) the exact same way as in Section 7. For any $j \in \{0, 1, \ldots, m-1\}$, $C \in V(F_j)$ and $\mathbf{a}_i \in A$, we also define the notion of *$\mathbf{a}_i$-heavy* and *originally $\mathbf{a}_i$-heavy* the same way as in Section 7. The only differences here is that $m = |V(F)| - 10g$ instead of $|V(F)|$.

▶ **Lemma 32.** *Let $j \in \{0, 1, \ldots, m\}$. Let $C \in V(F_j)$ be a vertex of minimum degree. Then there exists $\mathbf{a}_i \in A$ such that C is $\mathbf{a}_i$-heavy in F_j.*

Proof. By the construction, $|V(F_j)| \geq 10g$. Therefore by Lemma 29, we have $\deg_{F_j}(C) \leq 7$. Therefore by Lemma 31, we get $z_j(C, A) \geq 2 - 7/(1000 \cdot a \cdot g) \geq 1$. This implies that there exists $\mathbf{a}_i \in A$ such that $z_j(C, \{\mathbf{a}_i\}) \geq 1/a$. ◀

▶ **Lemma 33.** *For any $j \in \{0, \dots, m\}$ and for any $v \in \Gamma \cap P_j$, we have $\deg_{P_j}(v) \leq 8$.*

Proof. A similar argument as in the planar case applies here. The only difference here is that the minimum degree is at most 7. Therefore, every vertex can be the parent of at most 7 other vertices and can have at most one parent. ◀

▶ **Lemma 34.** *The inductive invariant (I1) is maintained.*

Proof. The exact same argument as in Section 7 applies here. ◀

Now we construct a forest T' on G' the same way as in Section 7.

▶ **Lemma 35.** *T' has at most $a + 10g$ connected components.*

Proof. If $|V(F)| \leq 10g$, then we are done. Otherwise, we have $|F_m| = 10g$. Now since $|A| = a$, by the construction, we have that the number of connected components of T' is at most $a + 10g$. ◀

Phase 5. We construct a forest T in G the exact same way as in Section 7, by setting $E(T) = E(T') \cup \bigcup_{C \in V(F)} E(T_C)$.

This completes the description of the algorithm.

8.1 Analysis

▶ **Lemma 36.** *T is a spanning forest in G, with at most $O(a + g)$ connected components.*

Proof. For any $C \in V(F)$, let g_C be the genus of $\Gamma[C]$. By Theorem 28 we know that the number of connected components in T_C is at most g_C. Therefore, by Lemma 35 we have that the number of connected components in T is at most $a + 10g + \sum_{C \in V(F)} g_C \leq a + 11g$. ◀

For the thinness of T, we follow a similar approach as in the planar case. There are two main differences here: First for any $j \in \{0, 1, \dots, m\}$, by Lemma 30 we have that the average degree of F_j is at most $20g$. Second a cut U is tiny if $z(\delta(U)) < 1/(1000 \cdot a \cdot g)$.

Let U be a cut in G. Similar to the planar case, we partition $E(T) \cap \delta(U)$ into three subsets:

1. $T_1 = \{\{\mathbf{a}_i, v\} \in E(T) \cap \delta(U) : \mathbf{a}_i \in A, v \in V(\Gamma)\}$.
2. $T_2 = \{\{u, v\} \in E(T) \cap \delta(U) : u \text{ and } v \text{ are in the same component of } \Gamma'\}$.
3. $T_3 = \{\{u, v\} \in E(T) \cap \delta(U) : u \text{ and } v \text{ are in different components of } \Gamma'\}$.

Also similar to the planar case, we partition T_3 into three subsets:

1. $T_{31} = \{\{u, v\} \in T_3 : u \in U, v \notin U, \exists C_u, C_v \in V(F) \text{ s.t. } u \in C_u, v \in C_v, U \cap C_v \neq \emptyset\}$.
2. $T_{32} = \{\{u, v\} \in T_3 : u \in U, v \notin U, \exists C_u, C_v \in V(F) \text{ s.t. } u \in C_u, v \in C_v, U \cap C_u \neq \emptyset\}$.
3. $T_{33} = \{\{u, v\} \in T_3 : u \in U, v \notin U, \exists C_u, C_v \in V(F) \text{ s.t. } u \in C_u, v \in C_v, U \cap C_v = \emptyset, C_u \subseteq U\}$.

▶ **Lemma 37.** *For any index $i \in \{1, 2, 31, 32\}$, there exists a constant α_i such that $|T_i| \leq \alpha_i \cdot a \cdot g \cdot z(\delta(U))$.*

Proof. A similar argument as in Lemmas 23, 24 and 25 applies here. There are two differences here. First the definition of a tiny cut is different. Second, for each $C \in V(F)$, C can be the parent of at most seven vertices and can have at most one parent. Therefore we get $|T_1| \leq 1000 \cdot a \cdot g \cdot z(\delta(U))$, $|T_2| \leq 1000 \cdot a \cdot g \cdot z(\delta(U))$, $|T_{31}| \leq 8000 \cdot a \cdot g \cdot z(\delta(U))$ and $|T_{32}| \leq 8000 \cdot a \cdot g \cdot z(\delta(U))$. ◀

▶ **Lemma 38.** *There exists a constant α_{33} such that $|T_{33}| \leq \alpha_{33} \cdot a \cdot g \cdot z(\delta(U))$.*

Proof. Let $e = \{u, v\} \in T_{33}$. We follow a similar approach as in the planar case. We define U_1, U_2, U_1', U_2', B, $D^{\text{in}}_{U_1'}$, $D^{\text{out}}_{U_1'}$, $D^{\text{in}}_{U_2'}$ and $D^{\text{out}}_{U_2'}$ the exact same way as in Lemma 25. Let $V_1 = \{C \in D^{\text{in}}_{U_1'} : \deg_{D^{\text{in}}_{U_1'}}(C) \geq 98g\}$, $V_2 = \{C \in D^{\text{in}}_{U_1'} : \deg_{D^{\text{in}}_{U_1'}}(C) \leq 20g\}$, $X_1 = \{C \in D^{out}_{U_2'} : \deg_{D^{out}_{U_2'}}(C) \geq 98g\}$, and $X_2 = \{C \in D^{out}_{U_2'} : \deg_{D^{out}_{U_2'}}(C) \leq 5g\}$. By Lemma 30 we have that $|V_1| \leq |V_2|$ and $|X_1| \leq |X_2|$. With these definitions, the rest of the proof is the same as in Lemma 25, and thus we get $|T_{33}| \leq 10000 \cdot a \cdot g \cdot z(\delta(U))$. ◀

▶ **Lemma 39.** *T is a $O(a \cdot g)$-thin spanning forest in G, with at most $O(a + g)$ connected components.*

Proof. By combining Lemmas 36, 37 and 38 we get $|T| \leq 24000 \cdot a \cdot g \cdot z(\delta(U))$, which proves the assertion. ◀

We are now ready to prove the main result of this Section.

▶ **Theorem 40.** *Let $a, g \geq 1$. Let G be a graph and $A \subseteq V(G)$, with $|A| = a$, such that $H = G \setminus A$ is a graph of genus g. Let $z : E(G) \to \mathbb{R}_{\geq 0}$ be β-thick for some $\beta > 0$. Then there exists a polynomial time algorithm which given G, A, an embedding of H on a surface of genus g, and z outputs a $O((a \cdot g)/\beta)$-thin spanning forest in G (w.r.t. z) with at most $O(a + g)$ connected components.*

Proof. For $\beta \geq 2$, by Lemma 39, we can find a $(24000 \cdot a \cdot g)$-thin spanning forest with at most $a + 11g$ connected components. For $0 < \beta < 2$, the claim follows by scaling z by a factor of $2/\beta$. ◀

9 Thin subgraphs in nearly-embeddable graphs

In this section we give our algorithm for computing a thin subgraph in a $(a, g, 1, p)$-nearly embeddable graph, proving Lemma 8. We first handle graphs without any apices, and then proceed to the general case.

9.1 $(0, g, 1, p)$-nearly embeddable graphs

We now describe our algorithm for computing a thin subgraph in a $(0, g, 1, p)$-nearly embeddable graph.

For the remainder of this subsection, let $\vec{G}$ be a $(0, g, 1, p)$-nearly embeddable digraph and let G be its symmetrization. Let $\vec{H}$ be the single vortex of $\vec{G}$ of width p, attached to some face $\vec{F}$ of $\vec{G}$. Let H and F be the symmetrizations of $\vec{H}$ and $\vec{F}$ respectively. Let $\{B_v\}_{v \in V(F)}$ be a path-decomposition of H of width p. Let $\vec{W}$ be a closed walk in $\vec{G}$ visiting all vertices in $V(\vec{H})$ and let W be its symmetrization. Let $z : E(G) \to \mathbb{R}_{\geq 0}$ be α-thick, for some $\alpha \geq 2$, and $\vec{W}$-dense. Let G' be the graph obtained by contracting F to a single vertex v^* in $G \setminus H$.

Following [8] we introduce the following notation. For any $u, v \in V(G)$, a *ribbon* R between u and v is the set of all parallel edges $e = \{u, v\}$ such that for every $e, e' \in R$, there exists a homeomorphism between e and e' on the surface. Let R' be a set of parallel edges in G. We say that an edge $e \in R'$ is central if the total weight of edges on each side of e in R' (containing e), is at least $z(R')/2$.

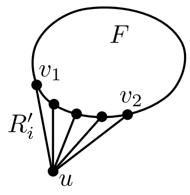
We will find a $O(1)$-thin spanning forest S in G' (w.r.t. z), with at most g connected components, such that S is $O(1)$-thin in G (w.r.t. z). We follow a similar approach to [8] to construct S. We apply some modifications that assure S is $O(1)$-thin in G (w.r.t. z).

9.1.1 The modified ribbon-contraction argument

If $|V(G')| \leq g$, then we set $E(S) = \emptyset$ and we are done. Otherwise, let $l = |V(G')| - g$. We define two sequences of graphs $G_0, \ldots, G_l$ and $G'_0, \ldots, G'_l$, with $G_0 = G$ and $G'_0 = G'$. For each $j \in \{0, \ldots, l\}$, G_j is obtained by uncontracting $v^* \in V(G'_j)$. Let $i \geq 0$ and suppose we have defined G'_i. Let R_i be the heaviest ribbon in G'_i (w.r.t. z). Let $R'_i \subseteq E(G_i)$ be the corresponding set of edges in G_i. We contract all the edges in R_i and we let G'_{i+1} be the graph obtained after contracting R_i. We also perform the contraction in a way such that for all $i \in \{0, \ldots, l-1\}$ we have $v^* \in G'_i$.

Let $i \in \{0, \ldots, l-1\}$. If $R_i = \{u, v\}$ where $u, v \neq v^*$, similar to [8], we let e_i be a central edge in R_i and we add e_i to S. Otherwise, suppose that $R_i = \{u, v^*\}$ for some $u \in V(G'_i)$. If there exists an edge $e \in R_i$ with $e \in W$ or $z(e) \geq 0.1$, we let $e_i = e$ and we add it to S. Otherwise, we can assume that there is no edge $e \in R$ with $e \in W$ or $z(e) \geq 0.1$.

Let Q_i be the set of vertices $v \in V(F)$ with an endpoint in R'_i. By the construction, Q_i is a subpath of F. Let $v_1, v_2 \in V(F)$ be the endpoints of Q_i. Let W'_i be the restriction of W on $\bigcup_{v \in Q_i} B_v$. Let W''_i be the subgraph of W'_i obtained by deleting all edges e with both endpoints in B_{v_1} or B_{v_2}.



For any subgraph C of W, we define the *i-load* of C as follows. The i-load of C is the total weight of all edges in R'_i with an endpoint in C. Let C_i be the connected component of W''_i with the maximum i-load. Let $Y_i = \{e \in R''_i :$ e has an endpoint in $C_i\}$. We let e_i be a central edge in Y_i and we add e_i to S.

We set $T = S \cup W$. We will show that T is a $O(p^2)$-thin spanning subgraph of G (w.r.t. z), with at most g connected components.

▶ **Lemma 41.** *T is a spanning subgraph of G with at most g connected components.*

Proof. By the construction, S has at most g connected components in G'. Now note that W is a closed walk visiting H in G. Therefore, all vertices of F are in the same connected component in T. This means that T has at most g connected components. ◀

▶ **Lemma 42.** *For any $i \in \{0, \ldots, l-1\}$, W'_i has at most $2p$ connected components.*

Proof. This follows immediately from the fact that $\{B_v\}_{v \in V(F)}$ is a path-decomposition of width p and there is no edge in $R_i \cap W$. ◀

▶ **Lemma 43.** *For any $i \in \{0, \ldots, l-1\}$, W''_i has at most $p(p+1)$ connected components.*

Proof. By Lemma 42 we know that W_i' has at most $2p$ connected components. W_i'' is obtained by deleting at most $p(p-1)$ edges of W_i'. Therefore, W_i'' has at most $p(p+1) = 2p + p(p-1)$ connected components. ◀

▶ **Lemma 44.** *For any $i \in \{0, \ldots, l-1\}$, the i-load of W_i' is at least 0.4.*

Proof. The i-load of W_i' is $z(R_i')$. Following [8] we know that $z(R_i) \geq 2/5$ and thus $z(R_i') \geq 2/5$. ◀

▶ **Lemma 45.** *For any $i \in \{0, \ldots, l-1\}$, the i-load of W_i'' is at least 0.2.*

Proof. By Lemma 44 the i-load of W' is at least 0.4. By the construction, we have $z(\{u, v_1\}) \leq 0.1$ and $z(\{u, v_2\}) \leq 0.1$. By deleting edges with both endpoints in B_{v_1} or B_{v_2}, we decrease the i-load by at most 0.2. Therefore, the i-load of W_i'' is at least 0.2. ◀

▶ **Lemma 46.** *For any $i \in \{0, \ldots, l-1\}$, the i-load of C_i is at least $1/5p(p+1)$.*

Proof. By Lemma 45 we know that the i-load of W_i'' is at least 0.2. By Lemma 43 there are at most $p(p+1)$ connected components in W_i''. Therefore the i-load of C_i is at least $1/5p(p+1)$. ◀

▶ **Lemma 47.** *There exists a constant β such that for any $U \subseteq V(G)$, we have $|S \cap \delta(U)| \leq \beta \cdot p^2 \cdot z(\delta(U))$.*

Proof. First we partition $S \cap \delta(U)$ into two subsets:

1. $S_1 = \{\{u, v\} \in S \cap \delta(U) : u, v \notin V(F)\}$.
2. $S_2 = \{\{u, v\} \in S \cap \delta(U) : v \in V(F)\}$.

By the construction, following [8] we have $|S_1| \leq 20 \cdot z(\delta(U))$. Let $e = \{u, v\} \in S_2$. Let $i \in \{0, \ldots, l-1\}$ be the step that we add e to S. If $e \in W$, we can charge it to $z(e) \geq 1/2$ and we are done. Suppose $e \notin W$. If there exists an edge $e' \in E(C_i) \cap \delta(U)$, we know that by the construction, e' does not have both endpoints in B_{v_1} or B_{v_2}. Therefore, for all $j \neq i$ we have $e' \notin E(C_j)$. Thus we can charge e to $z(e') \geq 1/2$ and we are done. Otherwise, suppose there is no edge in $E(C_i) \cap \delta(U)$. In this case, by the construction, for all $e'' \in R_i$ with an endpoint in C_i, we have $e'' \in \delta(U)$. Now we know that e is the central edge in Y_i. By Lemma 46 we know that the i-load of C_i is at least $1/5p(p+1)$. Therefore, we can charge e to the i-load of C_i and we get $|S_2| \leq 10 \cdot p^2 z(\delta(U))$. Therefore, we have $|S \cap \delta(U)| \leq 20 \cdot p^2 \cdot z(\delta(U))$. ◀

▶ **Lemma 48.** *Let $\vec{G}$ be a $(0, g, 1, p)$-nearly embeddable digraph, let $\vec{H}$ be its vortex, and let $\vec{W}$ be a walk in $\vec{G}$ visiting all vertices in $V(\vec{H})$. Let G, H, and W be the symmetrizations of $\vec{G}$, $\vec{H}$, and $\vec{W}$ respectively. Let $z : E(G) \to \mathbb{R}_{\geq 0}$ be α-thick for some $\alpha \geq 2$, and $\vec{W}$-dense. Then there exists a polynomial time algorithm which given $\vec{G}$, $\vec{H}$, $\vec{W}$, z, and an embedding of $\vec{G} \setminus \vec{H}$ into a surface of genus g, outputs a subgraph $S \subseteq G \setminus H$, satisfying the following conditions:*

1. *$W \cup S$ is a spanning subgraph of G and has $O(g)$ connected components.*
2. *$W \cup S$ is $O(p^2)$-thin w.r.t. z.*

Proof. The assertion follows immediately by Lemmas 41 and 47. ◀

9.2 $(a, g, 1, p)$-nearly embeddable graphs

For the remainder of this subsection, let $a, g, k, p \geq 0$ and $\vec{G}$ be an n-vertex $(a, g, 1, p)$-nearly embeddable digraph and let G be its symmetrization. Let $\vec{H}$ be the single vortex of width p, attached to a face $\vec{F}$ of $\vec{G}$. Let H and F be the symmetrization of $\vec{H}$ and $\vec{F}$ respectively. Let $A \subseteq V(G)$ with $|A| = a$ be the set of apices of G, where $\Gamma = G \setminus (A \cup H)$ is a graph of genus g. Let $\vec{W}$ be a walk in $\vec{G}$ visiting all vertices in $V(\vec{H})$ and let W be its symmetrization. Let $z : E(G) \to \mathbb{R}_{\geq 0}$ be α-thick, for some $\alpha \geq 2$, and $\vec{W}$-dense. Let G' be the graph obtained by contracting F to a single vertex v^* in $G \setminus H$.

We follow a similar algorithm as in Section 8 to find a $O(a \cdot g + p^2)$-thin spanning forest S in G', with at most $O(a + g)$ connected components. We modify the algorithm such that S is a $O(a \cdot g + p^2)$-thin subgraph of G.

We first start with Γ and construct Γ' the same way as in Section 8. For each connected component C of Γ', we want to find a $O(p^2)$-thin spanning forest T_C, with at most g connected components. Let C_{v^*} be the connected component of Γ' with $v^* \in C_{v^*}$. C_{v^*} is a graph of genus at most g. For this component, we apply the modified ribbon-contraction argument on Subsection 9.1 to find $T_{C_{v^*}}$. Therefore, $T_{C_{v^*}}$ is a $O(p^2)$-thin spanning forest in C_{v^*} with at most g connected components. The rest of the algorithm is the same as in Section 8 and we find a $O(a \cdot g + p^2)$-thin spanning forest S in G', with at most $O(a + g)$ connected components. Let $T = S \cup W$.

▶ **Lemma 49.** *T is a spanning subgraph of G with at most $O(a + g)$ connected components.*

Proof. The same proof as in Lemma 41 applies here. The only difference here is that S has at most $O(a + g)$ connected components in G'. Therefore, T has at most $O(a + g)$ connected components in G. ◀

▶ **Lemma 50.** *S is a $O(a \cdot g + p^2)$-thin subgraph of G (w.r.t. z).*

Proof. Let $U \subseteq V(G)$ be a cut. Similar to Subsection 9.1, we partition $S \cap \delta(U)$ into two subsets.

1. $S_1 = \{\{u, v\} \in S \cap \delta(U) : u, v \notin V(F)\}$.
2. $S_2 = \{\{u, v\} \in S \cap \delta(U) : v \in V(F)\}$.

First, by the construction and Lemma 39, we have $|S_1| \leq 24000 \cdot a \cdot g \cdot z(\delta(U))$. Now we partition S_2 into three subsets.

1. $S_{21} = \{\{a_j, v\} \in S_2 : a_j \in A, v \in V(F)\}$.
2. $S_{22} = \{\{u, v\} \in S_2 : v \in V(F), u$ and v are in different components of $\Gamma'\}$.
3. $S_{23} = \{\{u, v\} \in S_2 : v \in V(F), u, v \in C_{v^*}\}$.

By the construction, we know that $|S_{21}| \leq 1$ and $|S_{22}| \leq 7$. Also, by Lemma 47 we have $|S_{23}| \leq 20 \cdot p^2 \cdot z'(\delta(U))$. Therefore, we have $|S \cap \delta(U)| \leq 8(24000 \cdot a \cdot g + 20p^2)z(\delta(U))$. ◀

▶ **Lemma 51.** *T is a $O(a \cdot g + p^2)$-thin subgraph of G (w.r.t. z).*

Proof. By Lemma 50 we know that S is a $O(a \cdot g + p^2)$-thin subgraph of G (w.r.t. z). Now note that z is $\vec{W}$-dense. Therefore, $T = S \cup W$ is a $O(a \cdot g + p^2)$-thin subgraph of G (w.r.t. z). ◀

We are now ready to prove the main result of this Section.

Proof of Lemma 8. It follows by Lemmas 49 and 51. ◀

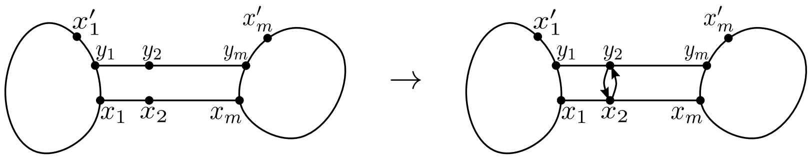
10 A preprocessing step for the dynamic program

▶ **Definition 52** (Facial normalization). Let $g \geq 0$, $p \geq 1$ and let $\vec{G}$ be a $(0, g, 1, p)$-nearly embeddable graph. Let $\vec{F}$ be the face on which the vortex is attached. We say that $\vec{G}$ is *facially normalized* if the symmetrization of $\vec{F}$ is a simple cycle and every $v \in V(\vec{F})$ has at most one incident edge that is not in $E(\vec{F})$.

▶ **Lemma 53.** *Let $g \geq 0$, $p \geq 1$ and let $\vec{G}$ be a $(0, g, 1, p)$-nearly embeddable graph. There exists a polynomial-time computable $(0, g, 1, p)$-nearly embeddable facially normalized graph $\vec{G}'$ such that the following holds. Let $\vec{H}$ be the vortex in $\vec{G}$ and let $\vec{H}'$ be the vortex in $\vec{G}'$. Then $\mathsf{OPT}_{\vec{G}}(V(\vec{H})) = \mathsf{OPT}_{\vec{G}'}(V(\vec{H}'))$. Moreover there exists a polynomial-time algorithm which given any closed walk $\vec{W}'$ in $\vec{G}'$ that visits all vertices in $V(\vec{H}')$, outputs some closed walk $\vec{W}$ in $\vec{G}$ that visits all vertices in $V(\vec{H})$ with $\mathsf{cost}_{\vec{G}}(\vec{W}) = \mathsf{cost}_{\vec{G}'}(\vec{W}')$.*

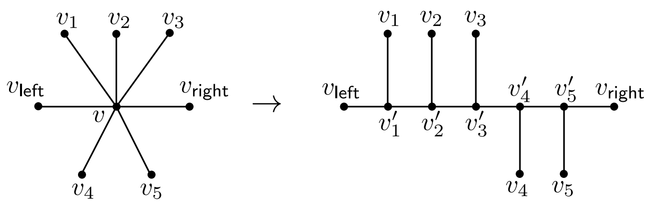
Proof. Let $\vec{F}$ be the face in $\vec{G}$ that $\vec{H}$ is attached to. Let F be the symmetrization of $\vec{F}$. We first construct a $(0, g, 1, p)$-nearly embeddable graph $\vec{G}''$, with a vortex $\vec{H}''$ attached to a face $\vec{F}''$ such that the symmetrization of $\vec{F}''$ is a simple cycle. Initially we set $\vec{G}'' = \vec{G}$. If $\vec{F}$ is a simple cycle then there is nothing to be done. Otherwise, suppose that $\vec{F}$ is not a simple cycle. Therefore, ∂F contains a family of simple cycles $\mathcal{C} = \{C_1, \ldots, C_k\}$ for some k, and a family of simple paths $\mathcal{P} = \{P_1, \ldots, P_l\}$ for some l, such that every $P_i \in \mathcal{P}$ is a path $x_1, x_2, \ldots, x_m$ where $x_1 \in C_\alpha$ and $x_m \in C_\beta$ for some $C_\alpha, C_\beta \in \mathcal{C}$. We allow $\mathcal{P}$ to contain paths of length 0.

For every $P = x_1, x_2, \ldots, x_m \in \mathcal{P}$, where $x_1 \in C$ and $x_m \in C'$ for some $C, C' \in \mathcal{C}$, we update $\vec{G}''$ as follows. Let $x'_1 \in V(C)$ and $x'_m \in V(C')$ be neighbors of x_1 and x_m. We first duplicate P to get a new path $P' = y_1, y_2, \ldots, y_m$. For every edge $e \in E(P)$, we set the cost of the corresponding edge $e' \in P'$ equal to the cost of e. Also, for every $j \in \{1, \ldots, m\}$, we add two edges (x_i, y_i) and (y_i, x_i) to $\vec{G}''$ with $\mathsf{cost}_{\vec{G}''}(x_i, y_i) = \mathsf{cost}_{\vec{G}''}(y_i, x_i) = 0$. Also, we delete edges (x_1, x'_1), (x'_1, x_1), (x_m, x'_m), and (x'_m, x_m) and we add edges (y_1, x'_1), (x'_1, y_1), (y_m, x'_m) and (x'_m, y_m) with the same cost respectively.



By the construction, $\vec{G}''$ is a $(0, g, 1, p)$-nearly embeddable graph, such that $\vec{F}''$ is a simple cycle. Also suppose that $\vec{W}''$ is a closed walk in $\vec{G}''$ that visits all vertices in $V(\vec{H}'')$. Then we can find a closed walk $\vec{W}$ in $\vec{G}$ that visits all vertices in $V(\vec{H})$ with $\mathsf{cost}_{\vec{G}}(\vec{W}) = \mathsf{cost}_{\vec{G}''}(\vec{W}'')$.

Now we construct a facially normalized graph $\vec{G}'$. Initially we set $\vec{G}' = \vec{G}''$. For every $v \in V(\vec{F}'')$ that has more than one incident edge in $E(\vec{G}'') \setminus E(\vec{F}'')$ we update $\vec{G}'$ as follows. Let $v_{\mathsf{left}}, v_{\mathsf{right}} \in V(\vec{F}'')$ be the left and right neighbors of v on $\vec{F}''$. Let $\mathcal{V} = \{v_1, \ldots, v_m\}$ be the set of all neighbors of v in $V(\vec{G}'') \setminus V(\vec{F}'')$. Let $\mathcal{V}' = \{v'_1, \ldots, v'_m\}$. First we delete v from $\vec{G}'$ and we add $\mathcal{V}'$ to $V(\vec{G}')$. For every $(v, v_i) \in E(\vec{G}'')$ we add (v'_i, v_i) to $E(\vec{G}')$ with $\mathsf{cost}_{\vec{G}'}(v'_i, v_i) = \mathsf{cost}_{\vec{G}''}(v, v_i)$. Also, for every $j \in \{1, \ldots, m-1\}$, we add (v'_j, v'_{j+1}) and (v'_{j+1}, v'_j) to $E(\vec{G}')$ with $\mathsf{cost}_{\vec{G}'}(v'_j, v_{j+1}) = 0$. Finally we add $(v'_1, v_{\mathsf{left}})$, $(v_{\mathsf{left}}, v'_1)$, $(v'_m, v_{\mathsf{right}})$ and $(v_{\mathsf{right}}, v'_m)$ to $\vec{G}'$ with the same costs as in $\vec{G}''$.



It is immediate that $\vec{G}'$ is the desired graph. ◀

▶ **Lemma 54.** *Let $g \geq 0$, $p \geq 1$ and let $\vec{G}$ be a $(0, g, 1, p)$-nearly embeddable graph. There exists a polynomial-time computable $(0, g, 1, p)$-nearly embeddable facially normalized graph $\vec{G}''$ such that the following conditions hold:*

1. *Let $\vec{H}$ be the vortex in $\vec{G}$ and let $\vec{H}''$ be the vortex in $\vec{G}''$. Then $\mathsf{OPT}_{\vec{G}}(V(\vec{H})) = \mathsf{OPT}_{\vec{G}'}(V(\vec{H}''))$.*
2. *There exists a polynomial-time algorithm which given any closed walk $\vec{W}''$ in $\vec{G}''$ that visits all vertices in $V(\vec{H}'')$, outputs some closed walk $\vec{W}$ in $\vec{G}$ that visits all vertices in $V(\vec{H})$ with $\mathsf{cost}_{\vec{G}}(\vec{W}) = \mathsf{cost}_{\vec{G}''}(\vec{W}'')$.*
3. *Let $\vec{\Gamma}$ be the genus-g piece of $\vec{G}''$. Let $\vec{F}''$ be the face of $\vec{\Gamma}$ on which the vortex $\vec{H}''$ is attached. Then any $v \in V(\vec{\Gamma}) \setminus V(\vec{F}'')$ has degree at most 4.*
4. *There exists some closed walk $\vec{W}^*$ in $\vec{G}''$ that visits all vertices in $V(\vec{H}'')$, with $\mathsf{cost}_{\vec{G}''}(\vec{W}^*) = \mathsf{OPT}_{\vec{G}''}(V(\vec{H}''))$, and such that every edge in $\vec{\Gamma}$ is traversed at most once by $\vec{W}^*$.*

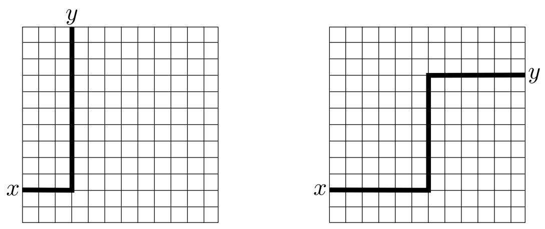
We say that a graph $\vec{G}''$ satisfying the above conditions is cross normalized.

Proof. We begin with computing the facially normalized graph $\vec{G}'$ given by Lemma 53. Clearly $\vec{G}'$ satisfies conditions (1) and (2).

We next modify $\vec{G}'$ so that it also satisfies (3). This can be done as follows. Let $\vec{\Gamma}'$ be the genus-g piece of $\vec{G}'$ and let $\vec{F}'$ be the face on which the vortex is attached. We replace each $v \in V(\vec{\Gamma}') \setminus V(\vec{F}')$ of degree $d > 4$ by a tree T_v with d leaves and with maximum degree 4; we replace each edge incident to v an edge incident to a unique leaf, and we set the length of every edge in $E(T_v)$ to 0.

It remains to modify $\vec{G}'$ so that it also satisfies (4). Let $\vec{H}'$ be the vortex in $\vec{G}'$. Let $\vec{W}'$ be a walk in $\vec{G}'$ that visits all vertices in $\vec{H}'$ with $\mathsf{cost}_{\vec{G}'}(\vec{W}') = \mathsf{OPT}_{\vec{G}'}(V(\vec{H}'))$. We may assume w.l.o.g. that $\vec{W}'$ contains at most n^2 edges. Thus, every vertex in $v \in V(\vec{\Gamma}') \setminus V(\vec{F}')$ is visited at most n^2 times by $\vec{W}'$. We replace each $v \in V(\vec{\Gamma}') \setminus V(\vec{F}')$ by a grid A_v of size $3n^2 \times 3n^2$, with each edge having length 0. Each edge incident to v in $\vec{G}'$, corresponds to a unique sides of A_v so that the ordering of the sides agrees with the ordering of the edges around v (in ψ). We replace each $(u, v) \in E(\vec{\Gamma}')$, by a matching of size $3n^2$ between the corresponding sides of A_u and A_v, where each edge in the matching has length equal to the length of (u, v). Let $\vec{G}''$ be the resulting graph.

We obtain the desired walk $\vec{W}^*$ in $\vec{G}''$ as follows. Let $x_1, \ldots, x_{4\ell}$ be the vertices in the boundary of A_v, with $\ell = 3n^2 - 1$, appearing in this order along a clockwise traversal of A_v, and such that x_1 is the lower left corner. Then for any $i \in \{1, \ldots, 4\}$, the vertices $x_{(i-1)\ell+1}, \ldots, x_{(i-1)\ell+n^2}$ correspond to the copies of v on the i-th side of A_v. We traverse $\vec{G}'$ starting at some arbitrary vertex in $V(\vec{H}')$, and we inductively construct the walk $\vec{W}^*$. We consider each edge (u, v) in the order that it is traversed by $\vec{W}'$. Suppose that (u, v) is the t-th edge traversed by $\vec{W}'$, for some $t \in \{1, \ldots, n^2\}$. For each $i \in \{1, \ldots, 4\}$, we let the t-th copy of v on the i-th side of A_v to be $x_{(i-1)\ell+t}$. We distinguish between the following cases: (i) If $u, v \in V(\vec{H}')$, then $(u, v) \in E(\vec{G}'')$ and we simply traverse (u, v) in $\vec{G}''$. (ii) If $u \in V(\vec{H}')$ and $v \notin V(\vec{H}')$ then we traverse the edge in $\vec{G}''$ that connects u to the t-th copy of v in the appropriate side of A_v. (iii) If $u \notin V(\vec{H}')$ and $v \in V(\vec{H}')$ then we traverse the edge in $\vec{G}''$ that connects the t-th copy of u in the appropriate side of A_u to v. (iv) If $u, v \notin V(\vec{H}')$ then we traverse the edge in $\vec{G}''$ that connects the t-th copy of u to the t-th copy of v in the appropriate sides of A_u and A_v respectively. Finally, for any pair of consecutive edges (u, v), (v, w) traversed by $\vec{W}'$, with $v \notin V(\vec{H}')$, we need to add a path P in A_v connecting two copies of v in the corresponding sides of A_v. Since A_v is a grid of size $3n^2 \times 3n^2$ this can be done so that all these paths are edge-disjoint. More precisely, this can be done as follows. Suppose that (u, v) is the t-th edge traversed by $\vec{W}'$, for some $t \in \{1, \ldots, n^2\}$. If P connects vertices x and y in consecutive sides of A_v, then we proceed as follows. We may assume w.l.o.g that $x = x_t$ and $y = x_{\ell+t+1}$, since other cases can be handled in a similar way. We set P to be the unique path starting at x_t, following $t + 1$ horizontal edges in A_v, and finally following $\ell - t$ vertical edges to $x_{\ell+t+1}$. Otherwise, if P connects vertices x and y in opposite sides of A_v, then we proceed as follows. We may assume w.l.o.g that $x = x_t$ and $y = x_{2\ell+t+1}$, since other cases can be handled in a similar way. We set P to be the unique path starting at x_t, following $t + n^2$ horizontal edges in A_v, and then following $\ell - 2t$ vertical edges, and finally following $\ell - t - n^2 - 1$ horizontal edges to $x_{2\ell+t+1}$.



By the construction, it is immediate that all the paths P constructed above are pairwise edge-disjoint, which implies that every edge in $E(\vec{G}'')$ is visited by $\vec{W}^*$ at most once, concluding the proof. ◀

11 Uncrossing an optimal walk traversing a vortex

Let $g \geq 0$, $p \geq 1$ and let $\vec{G}$ be a $(0, g, 1, p)$-nearly embeddable graph. By Lemma 54 we may assume w.l.o.g. that $\vec{G}$ is facially normalized and cross normalized. Let $\vec{G}' \subseteq \vec{G}$ be the piece of genus g and fix a drawing ψ of $\vec{G}'$ into a surface of genus g. Let $\vec{H}$ be the single vortex in $\vec{G}$ and suppose that $\vec{H}$ is attached to some face $\vec{F}$ of $\vec{G}'$. Fix an optimal solution $\vec{W}_{\mathsf{OPT}}$, that is a closed walk in $\vec{G}$ that visits all vertices in $\vec{H}$ minimizing $\mathsf{cost}_{\vec{G}}(\vec{W})$; if there are multiple such walks pick consistently one with a minimum number of edges. Since $\vec{G}$ is cross normalized we may assume w.l.o.g. that $\vec{W}_{\mathsf{OPT}}$ traverses every edge in $E(\vec{G}') \setminus E(\vec{F})$ at most once.

11.1 The structure of an optimal solution

▶ **Definition 55** (Shadow). Let $\mathcal{W}$ be a collection of walks in $\vec{G}$. We define *shadow* of $\mathcal{W}$ (w.r.t. $\vec{G}'$) to be the collection of open and closed walks obtained by restricting every walk in $\mathcal{W}$ on $\vec{G}'$ (note that a walk in $\mathcal{W}$ can give rise to multiple walks in $\mathcal{W}'$, and every open walk in $\mathcal{W}'$ must have both endpoints in $\vec{F}$).

We say that two edje-disjoint paths P, P' in $\vec{G}'$ *cross* (w.r.to ψ) if there exists $v \in V(P) \cap V(P')$ such that v has degree 4 (recall that $\vec{G}$ is cross normalized), with neighbors $u_1, \ldots, u_4$, such that the edges $\{v, u_1\}, \ldots, \{v, u_4\}$ appear in this order around v in the embedding ψ, P contains the subpath u_1, v, u_3, and P' contains the subpath u_2, v, u_4. We say that two walks in $\vec{G}'$ cross (at v, w.r.to ψ) if they contain crossing subpaths. Finally, a walk is *self-crossing* if it contains two disjoint crossing subpaths.

▶ **Lemma 56** (Uncrossing an optimal walk of a vortex). *There exists a collection* $\mathcal{W} = \{\vec{W}_1, \ldots, \vec{W}_\ell\}$ *of closed walks in* $\vec{G}$ *satisfying the following conditions:*

1. *Every edge in* $E(\vec{G}') \setminus E(\vec{F})$ *is traversed in total at most once by all the walks in* $\mathcal{W}$.
2. $V(\vec{W}_1) \cup \ldots \cup V(\vec{W}_\ell)$ *is a strongly-connected subgraph of* $\vec{G}$.
3. $V(\vec{H}) \subseteq V(\vec{W}_1) \cup \ldots \cup V(\vec{W}_\ell)$.
4. $\sum_{i=1}^{\ell} \mathsf{cost}_{\vec{G}}(\vec{W}_i) \leq \mathsf{OPT}_{\vec{G}}(V(\vec{H}))$.
5. *Let* $\mathcal{W}'$ *be the shadow of* $\mathcal{W}$. *Then the walks in* $\mathcal{W}'$ *are non-self-crossing and pairwise non-crossing.*

Proof. Initially, we set $\mathcal{W} = \{\vec{W}_{\mathsf{OPT}}\}$. Recall that since $\vec{G}$ is cross normalized, every edge in $E(\vec{G}') \setminus E(\vec{F})$ is traversed at most once by $\vec{W}_{\mathsf{OPT}}$. Clearly, this choice of $\mathcal{W}$ satisfies conditions (1)–(4). We proceed to iteratively modify $\mathcal{W}$ until condition (4) is also satisfied, while inductively maintaining (1)–(4).

Suppose that the current choice for $\mathcal{W}$ does not satisfy (5). This means that either there exist two distinct crossing walks in $\mathcal{W}$, or there exists some self-crossing walk in $\mathcal{W}$. In either case, it follows that there exist subpaths P, P' of the walks in $\mathcal{W}$ that are crossing (w.r.to ϕ). This means that there exists $v \in V(P) \cap V(P')$ and $e_1, e_2 \in E(P)$, $e_3, e_4 \in E(P')$ such that $\psi(e_1)$, $\psi(e_4)$, $\psi(e_2)$, $\psi(e_3)$ appear in this order around $\psi(v)$. We modify P and P' by swapping e_1 and e_3. It is immediate that the above operation preserves conditions (1)–(4). Moreover, after performing the operation, the total number of crossings and self-crossings (counted with multiplicities) between the walks in $\mathcal{W}$ decreases by at least one. Since the original number of crossings is finite, it follows that the process terminates after a finite number of iterations. By the inductive condition, it is immediate that when the process terminates the collection $\mathcal{W}$ satisfies condition (5), concluding the proof. ◀

For the remainder of this section let $\mathcal{W}$ and $\mathcal{W}'$ be as in Lemma 56. Let $\mathcal{I}$ be a graph with $V(\mathcal{I}) = \mathcal{W}'$ and with $E(\mathcal{I}) = \left\{\{W, W'\} \in \binom{\mathcal{W}'}{2} : V(W) \cap V(W') \neq \emptyset\right\}$.

Let $\vec{W}, \vec{Z}$ be distinct closed walks in some digraph, and let $v \in V(\vec{W}) \cap V(\vec{Z})$. Suppose that $\vec{W} = x_1, \ldots, x_k, v, x_{k+1}, \ldots, x_{k'}, x_1$ and $\vec{Z} = y_1, \ldots, y_r, v, y_{r+1}, \ldots, y_{r'}, y_1$. Let $\vec{S}$ be the closed walk $x_1, \ldots, x_k, v, y_{r+1}, \ldots, y_{r'}, y_1, \ldots, y_r, v, x_{k+1}, \ldots, x_{k'}, x_1$. We say that $\vec{S}$ is obtained by *shortcutting* $\vec{W}$ and $\vec{Z}$ (at v).

Let $\mathcal{J}$ be a subgraph of $\mathcal{I}$. Let $\mathcal{W}^{\mathcal{J}}$ be a collection of walks in $\vec{G}$ constructed inductively as follows. Initially we set $\mathcal{W}^{\mathcal{J}} = \mathcal{W}$. We consider all $\{W, W'\} \in E(\mathcal{J})$ in an arbitrary order. Note that since $\{W, W'\} \in E(\mathcal{J})$, it follows that W and W' cross at some $v \in V(\vec{G}')$. Let R and R' be the walks in $\mathcal{W}^{\mathcal{J}}$ such that W and W' are sub-walks of R and R' respectively. If $R \neq R'$ then we replace R and R' in $\mathcal{W}^{\mathcal{J}}$ by the walk obtained by shortcutting R and R' at v. This completes the construction of $\mathcal{W}^{\mathcal{J}}$. We say $\mathcal{W}^{\mathcal{J}}$ is obtained by *shortcutting* $\mathcal{W}$ *at* $\mathcal{J}$.

▶ **Lemma 57.** *There exists some forest* $\mathcal{F}$ *in* $\mathcal{I}$ *such that the collection of walks obtained by shortcutting* $\mathcal{W}$ *at* $\mathcal{F}$ *contains a single walk.*

Proof. Let $\mathcal{F}$ be a forest obtained by taking a spanning subtree in each connected component of $\mathcal{I}$. Let $W, W' \in \mathcal{W}$. Let $\mathcal{W}^{\mathcal{F}}$ be obtained by shortcutting $\mathcal{W}$ at $\mathcal{F}$. It suffices to show that W and W' become parts of the same walk in $\mathcal{W}^{\mathcal{F}}$. By condition (2) of Lemma 56 we have that there exists a sequence of walks $W_1, \ldots, W_t \in \mathcal{W}$, with $W_1 = W$, $W_t = W'$, and such that for any $i \in \{1, \ldots, t-1\}$, there exists some walk $A_i \in W_i \cap \vec{G}'$, and some walk $B_{i+1} \in W_{i+1} \cap \vec{G}'$ such that $\{A_i, B_{i+1}\} \in E(\mathcal{I})$. Therefore A_i and B_{i+1} are in the same connected component of $\mathcal{I}$. Thus there exits some tree $\mathcal{T}_i$ in $\mathcal{F}$ such that $A_i, B_{i+1} \in V(\mathcal{T}_i)$. It follows that after shortcutting $\mathcal{W}$ at $\mathcal{F}$, the walks A_i and B_{i+1} become parts of the same walk. By induction on $i \in \{1, \ldots, t-1\}$, it follows that A_1 and B_t become parts of the same walk in $\mathcal{W}^{\mathcal{F}}$, and thus so do W and W', concluding the proof. ◀

For the remainder let $\mathcal{F}$ be the forest given by Lemma 57.

▶ **Lemma 58.** *All leaves of $\mathcal{F}$ intersect F.*

Proof. Suppose that there exists some leaf $\vec{W}$ of $\mathcal{F}$ with $V(\vec{W}) \cap V(F) = \emptyset$. Then simply removing $\vec{W}$ from $\mathcal{W}$ leaves a new collection of walks that visits all vertices in $V(\vec{H})$ and such that the union of all walks is a strongly-connected subgraph of $\vec{G}$. Thus after shortcutting all these walks we may obtain a new single walk $\vec{R}$ that visits all vertices in $V(\vec{H})$ with $\mathsf{cost}_{\vec{G}}(\vec{R}) \leq \mathsf{cost}_{\vec{G}}(\vec{W}_{\mathsf{OPT}})$ and with fewer edges than $\vec{W}_{\mathsf{OPT}}$, contradicting the choice of $\vec{W}_{\mathsf{OPT}}$. ◀

12 The dynamic program for traversing a vortex in a planar graph

For the remainder of this section let $\vec{G}$ be a n-vertex $(0, 0, 1, p)$-nearly embeddable graph (that is, planar with a single vortex). Let $\vec{H}$ be the vortex in $\vec{G}$ and suppose it is attached on some face $\vec{F}$. Fix an optimal solution $\vec{W}_{\mathsf{OPT}}$, that is a closed walk in $\vec{G}$ that visits all vertices in $\vec{H}$ minimizing $\mathsf{cost}_{\vec{G}}(\vec{W})$; if there are multiple such walks pick consistently one with a minimum number of edges. Let F be the symmetrization of $\vec{F}$. We present an algorithm for computing a walk traversing all vertices in $V(\vec{H})$ based on dynamic programming. By Lemma 54 we may assume w.l.o.g. that $\vec{G}$ is facially normalized and cross normalized.

Fix a path-decomposition $\{B_v\}_{v \in V(F)}$ of $\vec{H}$ of width p.

Let $\mathcal{S}$ be a collection of walks in $\vec{G}$. For any $v \in V(\vec{G})$ we denote by $\mathsf{in\text{-}degree}_{\mathcal{S}}(v)$ the number of times that the walks in $\mathcal{S}$ enter v; similarly, we denote by $\mathsf{out\text{-}degree}_{\mathcal{S}}(v)$ the number of times that the walks in $\mathcal{S}$ exit v. We define $\vec{G}[\mathcal{S}]$ to be the graph with $V(\vec{G}[\mathcal{S}]) = \bigcup_{W \in \mathcal{S}} V(W)$ and $E(\vec{G}[\mathcal{S}]) = \bigcup_{W \in \mathcal{S}} E(W)$.

12.1 The dynamic program

Let $\mathcal{P}$ be the set of all subpaths of F, where we allow allow for simplicity in notation that a path be closed. Let u, v be the endpoints of P. Let $\vec{H}_P = \vec{H}\left[\bigcup_{x \in V(P)} B_x\right]$. Let $\mathcal{C}_P$ be the set of all possible partitions of $B_u \cup B_v$. Let $\mathcal{D}_P^{\mathsf{in}} = \{0, \ldots, n\}^{B_u \cup B_v}$, $\mathcal{D}_P^{\mathsf{out}} = \{0, \ldots, n\}^{B_u \cup B_v}$, that is, every element of $\mathcal{D}_P^{\mathsf{in}} \cup \mathcal{D}_P^{\mathsf{out}}$ is a function $f : B_u \cup B_v \to \{0, \ldots, n\}$. Let $\mathcal{A} = V(F)^2$.

12.1.1 The dynamic programming table

The dynamic programming table is indexed by all pairs (P, ϕ) where $P \in \mathcal{P}$ and $\phi = (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p) \in \mathcal{I}_P$, where

$$\mathcal{I}_P = \mathcal{C}_P \times \mathcal{D}_P^{\mathsf{in}} \times \mathcal{D}_P^{\mathsf{out}} \times (\mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}) \times (V(\vec{G}) \cup \mathsf{nil}) \times (V(\vec{G}) \cup \mathsf{nil}) \times (V(\vec{G}) \cup \mathsf{nil}).$$

A *partial solution* is a collection of walks in $\vec{G}$.

We say that a partial solution $\mathcal{S}$ is *compatible* with (P, ϕ) if the following conditions are satisfied:

(T1) For every $x \in V(\vec{H}_P)$ there exists some walk in $\mathcal{S}$ that visits x. That is $V(\vec{H}_P) \subseteq \bigcup_{Q \in \mathcal{S}} V(Q)$.

(T2) If $a \neq \mathsf{nil}$ and $a \in \mathcal{A}$, let $a = (u', v')$. Let Q_1 be the shortest path from u' to l in $\vec{G}$. Let Q_2 be the shortest path from l to r in $\vec{G}$. Let Q_3 be the shortest path from r to v' in $\vec{G}$. Let Q_1^* be the walk from u' to v' obtained by the concatenation of Q_1, Q_2 and Q_3. Then Q_1^* is a sub-walk of some walk in $\mathcal{S}$. We refer to Q^* as the *grip* of $\mathcal{S}$, and in this case we say that it is an *unbroken grip.* If $a \in V(P)^2$ then we say that the unbroken grip is *closed* and otherwise we say that it is *open.* Otherwise, if $a \in (\mathcal{A} \times \mathcal{A})$, let $a = ((u_1', v_1'), (u_2', v_2'))$. Let Q_1' be the shortest path from u_1' to l in $\vec{G}$. Let Q_2' be the shortest path from l to v_1' in $\vec{G}$. Let Q_2^* be the path from u_1' to v_1' obtained by the concatenation of Q_1' and Q_2'. Let Q_1'' be the shortest path from u_2' to r in $\vec{G}$. Let Q_2'' be the shortest path from r to v_2' in $\vec{G}$. Let Q_3^* be the path from u_2' to v_2' obtained by the concatenation of Q_1'' and Q_2''. Then Q_2^* and Q_3^* are sub-walks of some walks in $\mathcal{S}$. We refer to (Q_2^*, Q_3^*) as the *broken grip* of $\mathcal{S}$.

(T3) If $a = \mathsf{nil}$ or $a \in \mathcal{A}$, then every open walk in $\mathcal{S}$ has both endpoints in $B_u \cup B_v$, except possibly for one walk $W \in \mathcal{S}$ that contains the grip as a sub-walk. If $a = (Q_1, Q_2) \in (\mathcal{A} \times \mathcal{A})$, then every open walk in $\mathcal{S}$ has both endpoints in $B_u \cup B_v$, except possibly for at most two walks $W_1, W_2 \in \mathcal{S}$ that contain Q_1 and Q_2 as sub-walks (note that Q_1 and Q_2 might be sub-walks of the same walk in $\mathcal{S}$).

(T4) For all $x \in B_u \cup B_v$ we have $f^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{S}}(x)$ and $f^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{S}}(x)$.

(T5) For any $x, y \in B_u \cup B_v$ we have that if x and y are in the same set of the partition C then they are in the same weakly-connected component of $\bigcup_{W \in \mathcal{S}} W$. Moreover for any $z \in V(\vec{H}_P)$ there exists $z' \in B_u \cap B_v$ such that z and z' are in the same weakly-connected component of $\bigcup_{W \in \mathcal{S}} W$.

12.1.2 Merging partial solutions

We compute the values of the dynamic programming table inductively as follows. Let $P, P_1, P_2 \in \mathcal{P}$ such that $E(P_1) \neq \emptyset$, $E(P_2) \neq \emptyset$, $E(P_1) \cap E(P_2) = \emptyset$, and $P = P_1 \cup P_2$. Let $u \in V(P_1)$, $w \in V(P_1) \cap V(P_2)$, $v \in V(P_2)$ such that u, w are the endpoints of P_1 and w, v are the endpoints of P_2. Let

$$\phi = (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p) \in \mathcal{I}_P,$$

$$\phi_1 = (C_1, f_1^{\mathsf{in}}, f_1^{\mathsf{out}}, a_1, l_1, r_1, p_1) \in \mathcal{I}_{P_1},$$

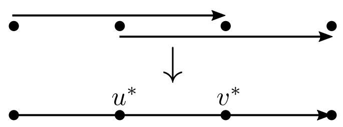
$$\phi_2 = (C_2, f_2^{\mathsf{in}}, f_2^{\mathsf{out}}, a_2, l_2, r_2, p_2) \in \mathcal{I}_{P_2}.$$

For any $i \in \{1, 2\}$ let $\mathcal{S}_i$ be a partial solution that is compatible with (P_i, ϕ_i). We proceed to compute a collection of walks $\mathcal{S}$ that is compatible with (P, ϕ). This is done in phases, as follows:

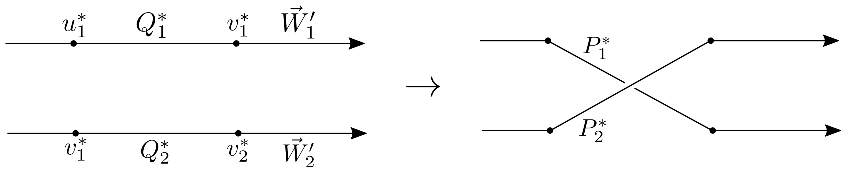
Merging phase 1: Joining the walks. We check that for all $x \in B_w$ we have $f_1^{\mathsf{in}}(x) = f_2^{\mathsf{out}}(x)$ and $f_2^{\mathsf{in}}(x) = f_1^{\mathsf{out}}(x)$. If not then the merging procedure return nil. For any $x \in B_w$ and for any $i \in \{1, 2\}$ let $E_i(x)^{\mathsf{in}}$ (resp. $E_i(x)^{\mathsf{out}}$) be the multiset of all edges in all walks in $\mathcal{S}_i$ that are incoming to (resp. outgoing from) x counted with multiplicities. Since $f_1^{\mathsf{out}}(x) = f_2^{\mathsf{in}}(x)$ and $f_2^{\mathsf{out}}(x) = f_1^{\mathsf{in}}(x)$ it follows that $|E_1^{\mathsf{in}}(x) \cup E_2^{\mathsf{in}}(x)| = |E_1^{\mathsf{out}}(x) \cup E_2^{\mathsf{out}}(x)|$. Pick an arbitrary bijection $\sigma_x : E_1^{\mathsf{in}}(x) \cup E_2^{\mathsf{in}} \to E_1^{\mathsf{out}}(x) \cup E_2^{\mathsf{out}}(x)$. We initially set $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$. For each $x \in B_w$ we proceed as follows. For each $e \in E_1^{\mathsf{in}(x)} \cup E_2^{\mathsf{in}}(x)$ we modify the walk traversing e so that immediately after traversing e it continues with the walk traversing $\sigma_x(e) \in E_1^{\mathsf{out}} \cup E_2^{\mathsf{out}}$.

Merging phase 2: Updating the grip. We check that at least one of the following conditions is satisfied:

1. Suppose that $a = l = r = p = a_1 = l_1 = r_1 = p_1 = a_2 = l_2 = r_2 = p_2 = \mathsf{nil}$. Then there is nothing to do.
2. Suppose that $a_1 = l_1 = r_1 = p_1 = \mathsf{nil}$, $l = l_2$, $r = r_2$, $p = p_2$ and $a = a_2 = (u_2^*, v_2^*) \in \mathcal{A}$ with $\{u_2^*, v_2^*\} \cap V(P_1) \subseteq \{u, w\}$, or $a_2 = l_2 = r_2 = p_2 = \mathsf{nil}$, $l = l_1$, $r = r_1$, $p = p_1$ and $a = a_1 = (u_1^*, v_1^*) \in \mathcal{A}$ with $\{u_1^*, v_1^*\} \cap V(P_2) \subseteq \{w, v\}$. Then there is nothing to do.
3. Suppose that $a_1 \neq \mathsf{nil}$, $a_2 \neq \mathsf{nil}$, and $a \neq \mathsf{nil}$. Suppose $a_1 = a_2 = a = (u^*, v^*) \in \mathcal{A}$, with $a \in V(P_1) \times V(P_2)$ or $a \in V(P_2) \times V(P_1)$. Suppose $l_1 = l_2 = l$, $r_1 = r_2 = r$ and $p_1 = p_2 = p$. Then we proceed as follows to ensure that (T2) holds. We may assume w.l.o.g. that $a \in V(P_1) \times V(P_2)$ since the remaining case can be handled in a similar way. Let Q^* be the grip between u^* and v^* in $\vec{G}$. It follows by (T2) that for any $i \in \{1, 2\}$ there exists a walk $\vec{W}_i \in \mathcal{S}_i$ that contains Q^* as a sub-walk. It follows by the definition of the merging phase 1 that for any $i \in \{1, 2\}$ there exists $\vec{W}_i' \in \mathcal{S}$ that contains Q^* as a sub-walk. We will modify $\mathcal{S}$ in order to ensure that (T2) holds. We remove Q^* from $\vec{W}_2'$ and we merge $\vec{W}_1'$ with $\vec{W}_2' \setminus Q^*$ (via concatenation).

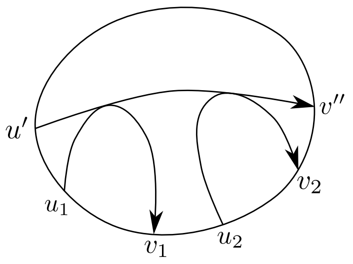


4. Suppose that $a_1, a_2, a \in \mathcal{A}$. Suppose that $a_1 = (u_1^*, v_1^*)$, $a_2 = (u_2^*, v_2^*)$, $a = (u^*, v^*)$, with $u^* \in \{u_1^*, u_2^*\}$, and $v^* \in \{v_1^*, v_2^*\}$. Suppose that $l_1 = l$, $l_2 = r_2 = r$ and $p_1 = p_2 = p$, or $l_2 = l$, $l_1 = r_1 = r$ and $p_1 = p_2 = p$, or $l = r = p_1 = p_2$ and $l_2 = r_2$, or $l = r = p_1 = p_2$ and $l_1 = r_1$. Then we proceed as follows to ensure that (T2) holds. We may assume w.l.o.g. that $a = (u_2^*, v_1^*)$, $l_1 = l$, $l_2 = r_2 = r$ and $p_1 = p_2 = p$ since the other cases can be handled in a similar way. For any $i \in \{1, 2\}$ let Q_i^* be the grip of $\mathcal{S}_i$. It follows by (T2) that for any $i \in \{1, 2\}$ there exists a walk $\vec{W}_i \in \mathcal{S}_i$ that contains Q_i^* as a sub-walk. It follows that for any $i \in \{1, 2\}$ there exists $\vec{W}_i' \in \mathcal{S}$ that contains Q_i^* as a sub-walk. We will modify $\mathcal{S}$ in order to ensure that (T2) holds. If $a = a_1$ or $a = a_2$ then there is nothing left to do. Otherwise, let R_1' be the shortest path in $\vec{G}$ from u_1^* to r_1. Let R_1'' be the shortest path in $\vec{G}$ from r_1 to l_2. Let R_1''' be the shortest path in $\vec{G}$ from l_2 to v_2^*. Let R_1^* be the path in $\vec{G}$ from u_1^* to v_2^* obtained by concatenation of R_1', R_1'' and R_1'''. Let R_2' be the shortest path in $\vec{G}$ from u_2^* to p. Let R_2'' be the shortest path in $\vec{G}$ from p to l_1. Let R_2''' be the shortest path in $\vec{G}$ from l_1 to v_1^*. Let R_2^* be the path in $\vec{G}$ from u_2^* to v_1^* obtained by concatenation of R_2', R_2'' and R_2'''. We remove Q_1^* and Q_2^* from $\vec{W}_1'$ and $\vec{W}_2'$ and replace them by R_1^* and R_2^*.



5. Suppose that $a_1 = \mathsf{nil}$ and $a = a_2 \in (\mathcal{A} \times \mathcal{A})$, or $a_2 = \mathsf{nil}$ and $a = a_1 \in (\mathcal{A} \times \mathcal{A})$. Then there is nothing to do.
6. Suppose that $a_1 \in \mathcal{A}$, $a_2 \in \mathcal{A}$ and $a \in (\mathcal{A} \times \mathcal{A})$. Suppose $a_1 = (u_1, v_1)$, $a_2 = (u_2, v_2)$ and $a = ((u', v'), (u'', v''))$, with $v' \in \{v_1, v_2\}$ and $u'' \in \{u_1, u_2\}$. We may assume w.l.o.g that $a = ((u', v_1), (u_2, v''))$. Suppose that $l = l_1 = r_1$, $r = l_2 = r_2$ and $p = p_1 = p_2$. For any $i \in \{1, 2\}$, let Q_i be the grip of $\mathcal{S}_i$. It follows by (T2) that for

any $i \in \{1,2\}$ there exists a walk $\vec{W}_i \in \mathcal{S}_i$ that contains Q_i as a sub-walk. It follows that for any $i \in \{1,2\}$ there exists $\vec{W}_i' \in \mathcal{S}$ that contains Q_i as a sub-walk. Let Q_1' be the shortest path in $\vec{G}$ from u_1 to l_1. Let Q_2' be the shortest path in $\vec{G}$ from l_1 to l_2. Let Q_3' be the shortest path in $\vec{G}$ from l_2 to v_2. Let Q_1^* be the path in $\vec{G}$ from u_1 to v_2 obtained by concatenation of Q_1', Q_2' and Q_3'. Let Q_1'' be the shortest path in $\vec{G}$ from u' to l_1. Let Q_2'' be the shortest path in $\vec{G}$ from l_1 to v_1. Let Q_2^* be the path in $\vec{G}$ from u' to v_1 obtained by concatenation of Q_1'' and Q_2''. Let Q_1''' be the shortest path in $\vec{G}$ from u_2 to l_2. Let Q_2''' be the shortest path in $\vec{G}$ from l_2 to v''. Let Q_3^* be the path in $\vec{G}$ from u_2 to v'' obtained by concatenation of Q_1''' and Q_2'''. Then we remove Q_1 and Q_2 from $\vec{W}_1'$ and $\vec{W}_2'$, and replace them by Q_1^*, Q_2^* and Q_3^*.



7. Suppose that $a_1 \in (\mathcal{A} \times \mathcal{A})$, $a_2 \in \mathcal{A}$ and $a \in (\mathcal{A} \times \mathcal{A})$, or $a_1 \in \mathcal{A}$, $a_2 \in (\mathcal{A} \times \mathcal{A})$ and $a \in (\mathcal{A} \times \mathcal{A})$. We may assume w.l.o.g that $a_1 \in (\mathcal{A} \times \mathcal{A})$, $a_2 \in \mathcal{A}$ and $a \in (\mathcal{A} \times \mathcal{A})$. Suppose that $a_1 = ((u_1, v_1), (u_1', v_1'))$, $a_2 = (u_2, v_2)$ and $a = ((u, v), (u', v'))$, with $(u, v) = (u_1, v_1)$, $u' \in \{u_1, u_2\}$, and $v' \in \{v_1, v_2\}$, or $(u, v) = (u_2, v_2)$, $u' \in \{u_1, u_2\}$, and $v' \in \{v_1, v_2\}$. We may assume w.l.o.g that $(u, v) = (u_1, v_1)$ and $(u', v') = (u_2, v_1')$. Suppose that $l = l_1$, $r = l_2 = r_2$ and $p = p_1 = p_2$. Let (Q_1, Q_1') be the grip of $\mathcal{S}_1$ and let Q_2 be the grip of $\mathcal{S}_2$. Let R_1 be the shortest path in $\vec{G}$ from u_1' to r_1. Let R_2 be the shortest path in $\vec{G}$ from r_1 to r_2. Let R_3 be the shortest path in $\vec{G}$ from r_2 to v_2. Let R' be the path in $\vec{G}$ from u_1' to v_2 obtained by concatenation of R_1, R_2 and R_3. Let R_1' be the shortest path in $\vec{G}$ from u_2 to r_2. Let R_2' be the shortest path in $\vec{G}$ from r_2 to v_1'. Let R'' be the path in $\vec{G}$ from u_2 to v_1' obtained by concatenation of R_1' and R_2'. Then we remove Q_1' and Q_2, and replace them by R' and R''.

If none of the above holds then the merging procedure returns nil.

Merging phase 3: Checking connectivity. We check that condition (T5) holds for $\mathcal{S}$ and we return nil if it does not.

▶ **Lemma 59.** *If the merging procedure outputs some partial solution $\mathcal{S}$ then $\mathcal{S}$ is compatible with ϕ.*

Proof. It follows immediately by the definition of compatibility. ◀

12.1.3 Initializing the dynamic programming table

For all $P \in \mathcal{P}$ containing at most one edge and for all $\phi \in \mathcal{I}_P$ with $\phi = (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p)$ we proceed as follows. We enumerate all partial solutions $\mathcal{S}$ that are compatible with (P, ϕ) and have minimum cost. Any walk in any such partial solution can intersect $\vec{G} \setminus \vec{H}$ only on the at most two oppositely-directed edges in $E(P)$. Moreover there are at most $O(n^7)$ possibilities for a, l, r and p. Thus the enumeration can clearly be done in time $n^{O(1)}$ by ensuring that for all walks $W \in \mathcal{S}$, their sub-walks that do not intersect $E(P)$ are shortest paths between vertices in $B_u \cup B_v$. The total running time of this initialization step is therefore $n^{O(p)}$.

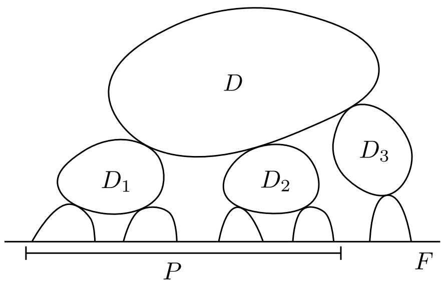


Figure 1 Example of a basic path.

12.1.4 Updating the dynamic programming table

For all $P \in \mathcal{P}$ containing $m > 1$ edges, and for all $P_1, P_2 \in \mathcal{P}$ with $E(P_1) \neq \emptyset$, $E(P_2) \neq \emptyset$, $E(P_1) \cap E(P_2) = \emptyset$ and $P_1 \cup P_2 = P$, and for all $\phi_1 \in \mathcal{I}_{P_1}$ and $\phi_2 \in \mathcal{I}_{P_2}$ we proceed as follows. Suppose that for all paths P' containing $m' < m$ edges and all $\phi' \in \mathcal{I}_{P'}$ we have computed the partial solutions in the dynamic programming table at (P', ϕ'). If there exist partial solutions $\mathcal{S}_1$ and $\mathcal{S}_2$ at (P_1, ϕ_1) and (P_2, ϕ_2) respectively, we call the merging process to merge $\mathcal{S}_1$ and $\mathcal{S}_2$. Suppose that the merging process returns a partial solution $\mathcal{S}$ at (P, ϕ) for some $\phi \in \mathcal{I}_P$. If there is no partial solution stored currently at (P, ϕ) then we store $\mathcal{S}$ at that location. Otherwise if there there exists a partial solution $\mathcal{S}'$ stored at (P, ϕ) and the cost of $\mathcal{S}$ is smaller than the cost of $\mathcal{S}'$ then we replace $\mathcal{S}'$ with $\mathcal{S}$.

12.2 Analysis

Let $\mathcal{W}$ be the collection of walks given by Lemma 56. Let $\mathcal{W}'$ be the shadow of $\mathcal{W}$. Let $\mathcal{F}$ be the forest obtained by Lemma 57. For every connected component $\mathcal{T}$ of $\mathcal{F}$ pick some $v_{\mathcal{T}} \in V(\mathcal{T})$ and consider $\mathcal{T}$ to be rooted at $v_{\mathcal{T}}$.

Let $\vec{G}'$ be the planar piece of $\vec{G}$, that is $\vec{G}' = \vec{G} \setminus (V(\vec{H}) \setminus (\vec{F}))$. Fix some planar drawing ψ of G'. Let $\mathcal{D}$ be the disk with $\partial\mathcal{D} = \psi(F)$ with $\psi(\vec{G}) \subset \mathcal{D}$.

Let $P \in \mathcal{P}$ with endpoints u, v, and let $\mathcal{T}$ be a subtree of some tree in $\mathcal{F}$. We say that P *covers* $\mathcal{T}$ if for all $D \in V(\mathcal{T})$ we have $V(D) \cap V(F) \subseteq V(P)$. We say that P *avoids* $\mathcal{T}$ if for all $D \in V(\mathcal{T})$ we have $V(D) \cap V(P) \subseteq \{u, v\}$.

▶ **Definition 60** (Basic path). Let $P \in \mathcal{P}$. Let $u, v \in V(P)$ be the endpoints of P. We say that P is *basic* (w.r.t. $\mathcal{W}$) if either $P \setminus \{u, v\}$ does not intersect any of the walks in $\mathcal{W}$ (in this case we call P *empty basic*) or the following holds. There exists some tree $\mathcal{T}$ in $\mathcal{F}$ and some $D \in V(\mathcal{T})$, with children $D_1, \ldots, D_k$, intersecting D in this order along a traversal of D, such that the following conditions are satisfied (see Figure 1):

1. For any $i \in \{1, \ldots, k\}$, let $\mathcal{T}_{D_i}$ be the subtree of $\mathcal{T}$ rooted at D_i and let $\mathcal{T}_D$ be the subtree of $\mathcal{T}$ rooted at D. Then at least one of the following two conditions is satisfied:
   1.1. P covers $\mathcal{T}_D$ and avoids $\mathcal{T} \setminus \mathcal{T}_D$.
   1.2. There exists $j \in \{1, \ldots, k\}$ such that for all $l \leq j$, P covers $\mathcal{T}_{D_l}$ and P avoids $\mathcal{T}_D \setminus \bigcup_{m=1}^{j} \mathcal{T}_{D_m}$.
2. Let $\mathcal{T}'$ be a tree in $\mathcal{F}$ with $\mathcal{T}' \neq \mathcal{T}$. Then either P covers $\mathcal{T}'$ or P avoids $\mathcal{T}'$.

▶ **Definition 61** (Facial restriction). Let $P \in \mathcal{P}$ be basic. Let $\mathcal{W}'$ be the collection of walks obtained by restricting every $W \in \mathcal{W}$ on H_P. If P is empty, we say that $\mathcal{W}'$ is the P*-facial restriction of* $\mathcal{W}$. Suppose that P is not empty. Let $\mathcal{T}$, D, $\mathcal{T}_D$, k and j be as in Definition 60.

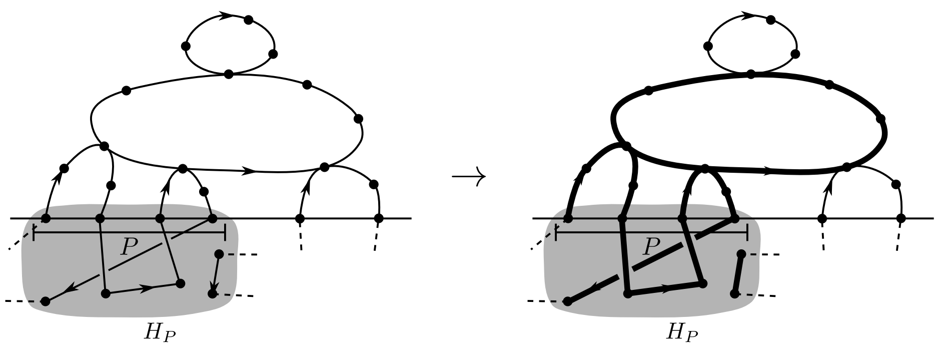


Figure 2 Part of the collection of walks in $\mathcal{W}$ (left) and the corresponding P-facial restriction of $\mathcal{W}$ depicted in bold (right).

Let $\mathcal{F}' = \{\mathcal{T}' \in \mathcal{F} : P \text{ covers } \mathcal{T}'\}$ and let $\mathcal{R} = \bigcup_{\mathcal{T}' \in \mathcal{F}'} V(\mathcal{T}')$. If P covers $\mathcal{T}_D$ and avoids $\mathcal{T} \setminus \mathcal{T}_D$, we say that $\mathcal{R} \cup \mathcal{W}' \cup V(\mathcal{T}_D)$ is the *P-facial restriction of $\mathcal{W}$*. Otherwise, we say that $\mathcal{R} \cup \mathcal{W}' \cup \{D\} \cup \bigcup_{i=1}^{j} V(\mathcal{T}_{D_i})$ is the *P-facial restriction of $\mathcal{W}$*. Figure 2 depicts an example of a P-facial restriction.

▶ **Definition 62** (Important walk)**.** Let $P \in \mathcal{P}$ be a basic path. Let $\mathcal{W}'$ be the P-facial restriction of $\mathcal{W}$. Let $\mathcal{W}''$ be the shadow of $\mathcal{W}'$. Let Q be a walk in $G[\mathcal{W}'']$. We say that Q is *P-important* (w.r.to. $\mathcal{W}$) if the following conditions hold:

1. Both endpoints of Q are in $V(P)$.
2. Q is the concatenation of walks $Q_1, \ldots, Q_\ell$ such that for each $i \in \{1, \ldots, \ell\}$ there exists some $W_i \in \mathcal{W}$ such that Q_i is a sub-walk of W_i, and for each $j \in \{1, \ldots, \ell-1\}$ we have $\{W_j, W_{j+1}\} \in E(\mathcal{F})$.

▶ **Proposition 63.** *For any $u, v \in V(P)$, there is at most one P-important walk from u to v.*

Proof. It follows immediately by the fact that $\mathcal{F}$ is a forest. ◀

▶ **Lemma 64.** *Let $P \in \mathcal{P}$ be basic w.r.t. $\mathcal{W}$ with endpoints $u, v \in V(F)$, where $u = v$ if P is closed. Let $\mathcal{W}_P$ be the P-facial restriction of $\mathcal{W}$. Let $\Gamma = \bigcup_{\vec{W} \in \mathcal{W}_P} \vec{W}$. Let C be the partition of $B_u \cup B_v$ that corresponds to the weakly-connected components of Γ. For any $x \in B_u \cup B_v$ let $f^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_P}(x)$ and $f^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_P}(x)$. Then there exists some $a \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}$ and $l, r, p \in (V(\vec{G}) \cup \mathsf{nil})$ such that the dynamic programming table contains some partial solution $\mathcal{S}$ at location $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$, such that $\mathsf{cost}_{\vec{G}}(\mathcal{S}) \le \mathsf{cost}_{\vec{G}}(\mathcal{W}_P)$.*

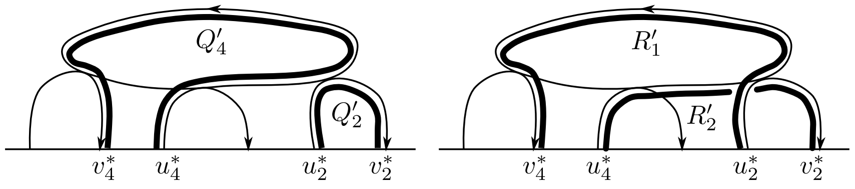
Proof. Let us first assume that $\mathcal{F}$ contains only one tree. We will deal with the more general case later on. First suppose that P is an empty basic path. Let $P = x_1, x_2, \ldots, x_m$, where $x_1 = u$ and $x_m = v$. We will prove the assertion by induction on m. For the base case, suppose that $m = 2$, and thus P contains only one edge. Let $a = l = r = p = \mathsf{nil}$. In this case, a partial solution $\mathcal{S}$ at location $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$ is computed in the initialization step of the dynamic programming table and clearly we have $\mathsf{cost}_{\vec{G}}(\mathcal{S}) \le \mathsf{cost}_{\vec{G}}(\mathcal{W}_P)$ and we are done. Now suppose that $m > 2$ and we have proved the assertion for all $m' < m$. We first decompose P into two edge-disjoint paths P_1 and P_2, such that $V(P_1) \cap V(P_2) = w$ for some $1 < j < m$ and $w = x_j$. For $i \in \{1, 2\}$ let $\mathcal{W}_{P_i}$ be the P_i-facial restriction of $\vec{W}_{\mathsf{OPT}}$ and let $\Gamma_i = \bigcup_{\vec{W} \in \mathcal{W}_{P_i}} \vec{W}$. Let C_1 be the partition of $B_u \cup B_w$ that corresponds to the weakly-connected components of Γ_1 and let C_2 be the partition of $B_w \cup B_v$ that corresponds to the weakly-connected components of Γ_2. For any $x \in B_u \cup B_w$ let $f_1^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_{P_1}}(x)$ and $f_1^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_{P_1}}(x)$. Also for any $x \in B_w \cup B_v$ let $f_2^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_{P_2}}(x)$ and $f_2^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_{P_2}}(x)$. By the induction hypothesis, there exists partial solutions $\mathcal{S}_1$ and $\mathcal{S}_2$ that are compatible with $(P_1, (C_1, f_1^{\mathsf{in}}, f_1^{\mathsf{out}}, \mathsf{nil}, \mathsf{nil}, \mathsf{nil}, \mathsf{nil}))$ and $(P_2, (C_2, f_2^{\mathsf{in}}, f^{\mathsf{out}}, \mathsf{nil}, \mathsf{nil}, \mathsf{nil}, \mathsf{nil}))$ respectively, and we have $\mathsf{cost}_{\vec{G}}(\mathcal{S}_1) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_1})$, $\mathsf{cost}_{\vec{G}}(\mathcal{S}_2) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_2})$. The dynamic program will merge $\mathcal{S}_1$ and $\mathcal{S}_2$ to get the desired $\mathcal{S}$. Note that by the construction, for every $x \in B_w$ we have $f_1^{\mathsf{in}}(x) = f_2^{\mathsf{out}}(x)$ and $f_2^{\mathsf{in}}(x) = f_1^{\mathsf{out}}(x)$. Therefore, by the first merging phase, we can merge walks in $\mathcal{S}_1$ and $\mathcal{S}_2$. Also, we let $a = l = r = p = \mathsf{nil}$ and we proceed the second phase of merging. Finally, by the construction and definition of P-facial restriction, (T5) holds and the merging process returns a partial solution $\mathcal{S}$ compatible with $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$ with $\mathsf{cost}_{\vec{G}}(\mathcal{S}) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_P)$, as desired.

Now suppose that P is not empty basic. Let $\mathcal{T}$, D, $\mathcal{T}_D$, k and j be as in Definition 60. We will prove the assertion by induction on $\mathcal{T}$. For the base case, suppose that D is a leaf of $\mathcal{T}$. Suppose that $P = x_1, x_2, \ldots, x_m$ for some $m > 0$, where $x_1 = u$ and $x_m = v$, and D is a (possibly closed) walk from $x_i \in V(P)$ to $x_j \in V(P)$. We may assume w.l.o.g. that $i \leq j$. There are some possible cases here based on i and j. First suppose that $i > 1$, $j < m$ and $j - i \geq 3$. In this case, let $P_1 = x_1, \ldots, x_i$, $P_2 = x_i, x_{i+1}$, $P_3 = x_{i+1}, \ldots, x_{j-1}$, $P_4 = x_{j-1}, x_j$ and $P_5 = x_j, \ldots, x_m$. Let $P_6 = P_1 \cup P_2$, $P_7 = P_6 \cup P_3$ and $P_8 = P_7 \cup P_4$. For $i \in \{1,2,3,4,5,6,7,8\}$ let $\mathcal{W}_{P_i}$ be the P_i-facial restriction of $\vec{W}_{\mathsf{OPT}}$ and let $\Gamma_i = \bigcup_{\vec{W} \in \mathcal{W}_{P_i}} \vec{W}$. We also define C_i, f_i^{in} and f_i^{out} as in the previous case. Note that P_1, P_3 and P_5 are empty basic paths. We let $a_1 = a_3 = a_5 = \mathsf{nil}$, $l_1 = l_3 = l_5 = \mathsf{nil}$, $r_1 = r_3 = r_5 = \mathsf{nil}$, $p_1 = p_3 = p_5 = \mathsf{nil}$ and thus we can find partial solutions $\mathcal{S}_1$, $\mathcal{S}_3$ and $\mathcal{S}_5$ compatible with $(P_1, (C_1, f_1^{\mathsf{in}}, f_1^{\mathsf{out}}, a_1, l_1, r_1, p_1))$, $(P_3, (C_3, f_3^{\mathsf{in}}, f_3^{\mathsf{out}}, a_3, l_3, r_3, p_3))$ and $(P_5, (C_5, f_5^{\mathsf{in}}, f_5^{\mathsf{out}}, a_5, l_5, r_5, p_5))$ respectively. We also have $\mathsf{cost}_{\vec{G}}(\mathcal{S}_1) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_1})$, $\mathsf{cost}_{\vec{G}}(\mathcal{S}_3) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_3})$ and $\mathsf{cost}_{\vec{G}}(\mathcal{S}_5) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_5})$. Let $a_2 = a_4 = a_6 = a_7 = a_8 = (x_i, x_j)$. If D does not have a parent in $\mathcal{T}$, then we let $l_2 = l_4 = l_6 = l_7 = l_8 = r_2 = r_4 = r_6 = r_7 = r_8 = p_2 = p_4 = p_6 = p_7 = p_8 \in V(D)$ to be an arbitrary vertex of D. Otherwise, suppose that D has a parent D' in $\mathcal{T}$. If D' does not have a parent in $\mathcal{T}$, then we let $l_2 = l_4 = l_6 = l_7 = l_8 = r_2 = r_4 = r_6 = r_7 = r_8 = p_2 = p_4 = p_6 = p_7 = p_8 \in V(D) \cap V(D')$. Otherwise, suppose that D' has a parent D'' in $\mathcal{T}$. In this case, we let $l_2 = l_4 = l_6 = l_7 = l_8 = r_2 = r_4 = r_6 = r_7 = r_8 \in V(D) \cap V(D')$ and $p_2 = p_4 = p_6 = p_7 = p_8 \in V(D') \cap V(D'')$. Therefore, by computing the initialization step, we can find a partial solution $\mathcal{S}_2$ compatible with $(P_2, (C_2, f_2^{\mathsf{in}}, f_2^{\mathsf{out}}, a_2, l_2, r_2, p_2))$ and a partial solution $\mathcal{S}_4$ compatible with $(P_4, (C_4, f_4^{\mathsf{in}}, f_4^{\mathsf{out}}, a_4, l_4, r_4, p_4))$, and we have $\mathsf{cost}_{\vec{G}}(\mathcal{S}_2) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_2})$ and $\mathsf{cost}_{\vec{G}}(\mathcal{S}_4) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_4})$. Now by merging $\mathcal{S}_1$ and $\mathcal{S}_2$, we get a partial solution $\mathcal{S}_6$ compatible with $(P_6, (C_6, f_6^{\mathsf{in}}, f_6^{\mathsf{out}}, a_6, l_6, r_6, p_6))$. By merging $\mathcal{S}_6$ and $\mathcal{S}_3$, we get a partial solution $\mathcal{S}_7$ compatible with $(P_7, (C_7, f_7^{\mathsf{in}}, f_7^{\mathsf{out}}, a_7, l_7, r_7, p_7))$. By merging $\mathcal{S}_7$ and $\mathcal{S}_4$, we get a partial solution $\mathcal{S}_8$ compatible with $(P_8, (C_8, f_8^{\mathsf{in}}, f_8^{\mathsf{out}}, a_8, l_8, r_8, p_8))$, and finally by merging $\mathcal{S}_8$ and $\mathcal{S}_5$, we get the desired partial solution $\mathcal{S}$ compatible with $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$ with $\mathsf{cost}_{\vec{G}}(\mathcal{S}) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_P)$. If $i = 1$ or $j = m$, we will follow a similar approach. The only different is that instead of dividing P into five paths, we divide it into four paths. Finally, the last case is when $j - i < 3$. In this case, if $i \neq j$, we define the same subpaths P_1, P_2, P_4 and P_5, and we follow a similar approach. Otherwise, suppose that $i = j$. In this case, we let $P_1 = x_1, \ldots, x_i$ and $P_2 = x_i, \ldots, x_m$ and by following the same approach by mering two partial solutions, we get the desired $\mathcal{S}$.

Now suppose that $D \in V(\mathcal{T})$ is non-leaf. In this case, we prove the assertion by induction on j, where j comes from Definition 60. Note that we perform a second induction inside the first induction. For the base case, suppose that $j = 1$. In this case, D_1 is a child of D and P covers $\mathcal{T}_{D_1}$ and avoids $\mathcal{T}_D \setminus \mathcal{T}_{D_1}$. Therefore, by using the first induction hypothesis on D_1, there exists $a \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}$ and $l, r, p \in V(\vec{G}) \cup \mathsf{nil}$ such that the dynamic programming table contains some partial solution $\mathcal{S}$ at location $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$ with $\mathsf{cost}_{\vec{G}}(\mathcal{S}) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_P)$. Now for the same a, l, r, p and $\mathcal{S}$, we have that $\mathcal{S}$ is compatible with $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$, as desired. Now suppose that we have proved the assertion for all $1 \leq j' < j$. By Definition 60, there exists a basic path $P_1 \subseteq P$, where u is the first vertex of P_1, such that for all $l \leq j-1$, P_1 covers $\mathcal{T}_{D_l}$ and avoids $\mathcal{T}_D \setminus (\bigcup_{m=1}^{j-1} \mathcal{T}_{D_m})$. Also there exists a basic path $P_2 \subseteq P$, where v is the last vertex of P_2, such that P_2 covers $\mathcal{T}_{D_j}$ and avoids $\mathcal{T} \setminus \mathcal{T}_{D_j}$. Let $u' \in V(F)$ and $v' \in V(F)$ be the other endpoints of P_1 and P_2 respectively. Let $P_3 \in \mathcal{P}$ be the path between u' and v' that does not contain v and let $P_4 = P_1 \cup P_3$. By the construction, P_3 and P_4 are basic. For $i \in \{1, 2, 3, 4\}$, let $\mathcal{W}_{P_i}$ be the P_i-facial restriction of $\vec{W}_{OPT}$ and let $\Gamma_i = \bigcup_{\vec{W} \in \mathcal{W}_{P_i}} \vec{W}$. Let C_1, C_2, C_3 and C_4 be the partitions of $B_u \cup B_{u'}$, $B_{v'} \cup B_v$, $B_{u'} \cup B_{v'}$ and $B_u \cup B_{v'}$ that corresponds to the weakly connected components of Γ_1, Γ_2, Γ_3 and Γ_4 respectively. For any $x \in B_u \cup B_{u'}$ let $f_1^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_{P_1}}(x)$ and $f_1^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_{P_1}}(x)$, for any $x \in B_{v'} \cup B_v$ let $f_2^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_{P_2}}(x)$ and $f_2^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_{P_2}}(x)$, for any $x \in B_{u'} \cup B_{v'}$, let $f_3^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_{P_3}}(x)$ and $f_3^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_{P_3}}(x)$, and for any $x \in B_u \cup B_{v'}$ let $f_4^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_{P_4}}(x)$ and $f_4^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_{P_4}}(x)$. By the second induction hypothesis, there exists some $a_1 \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}$ and $l_1, r_1, p_1 \in V(\vec{G}) \cup \mathsf{nil}$, such that the dynamic programming table contains some partial solution $\mathcal{S}_1$ at location $(P_1, (C_1, f_1^{\mathsf{in}}, f^{\mathsf{out}}, a_1, l_1, r_1, p_1))$, with $\mathsf{cost}_{\vec{G}}(\mathcal{S}_1) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_1})$. Also by the first induction hypothesis, there exists some $a_2 \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}$ and $l_2, r_2, p_2 \in V(\vec{G}) \cup \mathsf{nil}$, such that the dynamic programming table contains some partial solution $\mathcal{S}_2$ at location $(P_2, (C_2, f_2^{\mathsf{in}}, f^{\mathsf{out}}, a_2, l_2, r_2, p_2))$, with $\mathsf{cost}_{\vec{G}}(\mathcal{S}_2) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_2})$. Let $a_3 = l_3 = r_3 = p_3 = \mathsf{nil}$. Let $a_4 = a_1$, $l_4 = l_1$, $r_4 = r_1$ and $p_4 = p_1$. Since P_3 is basic, there exists a partial solution $\mathcal{S}_3$ compatible with $(P_3, (C_3, f_3^{\mathsf{in}}, f_3^{\mathsf{out}}, a_3, l_3, r_3, p_3))$. Now we merge $\mathcal{S}_1$ and $\mathcal{S}_3$ to get a partial solution $\mathcal{S}_4$ compatible with $(P_4, (C_4, f_4^{\mathsf{in}}, f_4^{\mathsf{out}}, a_4, l_4, r_4, p_4))$. Note that for every $x \in B_{u'}$, we have $f_3^{\mathsf{in}}(x) = f_1^{\mathsf{out}}(x)$ and $f_3^{\mathsf{out}}(x) = f_1^{\mathsf{in}}(x)$. Therefore, we can apply the first merging phase. Also we have $a_4 = a_1$, $l_4 = l_1$, $r_4 = r_1$ and $p_4 = p_1$, and thus we can apply the second merging phase. Finally, by the construction (T5) holds and we get a partial solution $\mathcal{S}_4$ compatible with $(P_4, (C_4, f_4^{\mathsf{in}}, f_4^{\mathsf{out}}, a_4, l_4, r_4, p_4))$. Now, we merge two partial solutions $\mathcal{S}_4$ and $\mathcal{S}_2$ to get the desired $\mathcal{S}$. Clearly, for every $x \in B_{v'}$ we have $f_4^{\mathsf{in}}(x) = f_2^{\mathsf{out}}(x)$ and $f_4^{\mathsf{out}}(x) = f_2^{\mathsf{in}}(x)$. Therefore, we can apply the first phase of merging. If $a_2 = l_2 = r_2 = p_2 = a_4 = l_4 = r_4 = p_4 = \mathsf{nil}$, then we let $a = l = r = p = \mathsf{nil}$. Otherwise, if $a_4 = l_4 = r_4 = p_4 = \mathsf{nil}$ and $a_2 = (u_2^*, v_2^*) \in \mathcal{A}$ with $\{u_2^*, v_2^*\} \cap V(P_4) \subseteq \{u, v'\}$, then we let $a = a_2$, $l = l_2$, $r = r_2$ and $p = p_2$. If $a_2 = l_2 = r_2 = p_2 = \mathsf{nil}$ and $a_4 = (u_4^*, v_4^*)$ with $\{u_4^*, v_4^*\} \cap V(P_2) \subseteq \{v', v\}$, then we let $a = a_4$, $l = l_4$, $r = r_4$ and $p = p_4$. Otherwise, if $a_2 \neq \mathsf{nil}$, $a_4 \neq \mathsf{nil}$, $a_4 = a_2$, $l_4 = l_2$, $r_4 = r_2$ and $p_4 = p_2$, then we let $a = a_4$, $l = l_4$, $r = r_4$ and $p = p_4$. Otherwise, if $a_2 = (u_2^*, v_2^*) \in \mathcal{A}$, $a_4 = (u_4^*, v_4^*) \in \mathcal{A}$, $l_4 = r_4$ and $p_1 = p_2$, then we let $a = (u^*, v^*)$, where $u^* \in \{u_2^*, u_4^*\}$ and $v^* \in \{v_2^*, v_4^*\}$, $l = l_2$, $r = r_4$ and $p = p_2$. Otherwise, if $a_2 = \mathsf{nil}$ and $a_4 \in (\mathcal{A} \times \mathcal{A})$, then we let $a = a_4$, $l = l_4$, $r = r_4$ and $p = p_4$. Otherwise, if $a_4 = \mathsf{nil}$ and $a_2 \in (\mathcal{A} \times \mathcal{A})$, then we let $a = a_2$, $l = l_2$, $r = r_2$ and $p = p_2$. Otherwise, if $a_2 = (u_2^*, v_2^*) \in \mathcal{A}$, $a_4 = (u_4^*, v_4^*) \in \mathcal{A}$, $l_2 = r_2$, $l_4 = r_4$ and $p_2 = p_4$, then we let $a = ((u', v'), (u'', v''))$ where $v' \in \{v_2, v_4\}$ and $u'' \in \{u_2, u_4\}$, $l = l_2$, $r = r_2$ and $p = p_2$. Otherwise, if $a_4 = ((u_4, v_4), (u_4', v_4')) \in (\mathcal{A} \times \mathcal{A})$, $a_2 = (u_2, v_2) \in \mathcal{A}$, $l_2 = r_2$ and $p_2 = p_4$, then we let $a = ((u_4, v_4), (u_2, v_4'))$, $l = l_4$, $r = r_2$ and $p = p_2$. Therefore, after applying the second merging phase, we get a partial solution $\mathcal{S}$ compatible with $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$.

Now we have to show that $\mathsf{cost}_{\vec{G}}(\mathcal{S}) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_P)$. Let us first suppose that D is a closed walk. We will deal with the case where D is an open walk later on. If $a_2 = l_2 = r_2 = p_2 = \mathsf{nil}$ or $a_4 = l_4 = r_4 = p_4 = \mathsf{nil}$, then this is immediate. If $a_2 = a_4$, $l_2 = l_4$, $r_2 = r_4$ and $p_2 = p_4$, then we have $a = a_2$, $l = l_2$, $r = r_2$ and $p = p_2$, and thus this case is also immediate. Suppose that $a_2 = (u_2^*, v_2^*) \neq \mathsf{nil}$, $a_4 = (u_4^*, v_4^*) \neq \mathsf{nil}$, $l_4 = r_4$ and $p_1 = p_2$, and $a = (u^*, v^*)$, where $u^* \in \{u_2^*, u_4^*\}$ and $v^* \in \{v_2^*, v_4^*\}$. We may assume w.l.o.g that $a = (u_2^*, v_4^*)$. We have that $l = l_2$, $r = r_4$ and $p = p_2$. For $i \in \{2,4\}$ let Q_i^* be the grip of $\mathcal{S}_i$, and let $\vec{W}_i \in \mathcal{S}_i$ that contains Q_i^* as a sub-walk. Let Y_1 be the shortest-path in $\vec{G}$ from u_2^* to l_2. Let Y_2 be the shortest-path in $\vec{G}$ from l_2 to l_4. Let Y_3 be the shortest-path in $\vec{G}$ from l_4 to v_4^*. Let R_1^* be the path in $\vec{G}$ from u_2^* to v_4^* obtained by concatenation of Y_1, Y_2 and Y_3. Let Y_1' be the shortest-path in $\vec{G}$ from u_4^* to r_4. Let Y_2' be the shortest-path in $\vec{G}$ from r_4 to l_2. Let Y_3' be the shortest-path in $\vec{G}$ from l_2 to v_2^*. Let R_2^* be the path in $\vec{G}$ from u_4^* to v_2^* obtained by concatenation of Y_1', Y_2' and Y_3'. Now by the construction, we have that $\mathsf{cost}_{\vec{G}}(\mathcal{S}) = \mathsf{cost}_{\vec{G}}(\mathcal{S}_2) + \mathsf{cost}_{\vec{G}}(\mathcal{S}_4) + \mathsf{cost}_{\vec{G}}(R_1^*) + \mathsf{cost}_{\vec{G}}(R_2^*) - \mathsf{cost}_{\vec{G}}(Q_2^*) - \mathsf{cost}_{\vec{G}}(Q_4^*)$. Note that by the construction, there exists a P-important walk from u_2^* to v_4^* (w.r.t. $\mathcal{W}$) and a P-important walk from u_4^* to v_2^* (w.r.t. $\mathcal{W}$). By Proposition 63 these walks are unique. Let R_1' be the P-important walk from u_2^* to v_4^* (w.r.t. $\mathcal{W}$) and let R_2' be the P-important walk from u_4^* to v_2^* (w.r.t. $\mathcal{W}$). Also, there exists a P_2-important walk from u_2^* to v_2^* (w.r.t. $\mathcal{W}$) and a P_4-important walk from u_4^* to v_4^* (w.r.t. $\mathcal{W}$), and thus by Proposition 63 these walks are unique. Let Q_2' be the P_2-important walk from u_2^* to v_2^* (w.r.t. $\mathcal{W}$) and let Q_4' be the P_4-important walk from u_4^* to v_4^* (w.r.t. $\mathcal{W}$). By the definition of important walks, we have that $\mathsf{cost}_{\vec{G}}(R_1^*) \leq \mathsf{cost}_{\vec{G}}(R_1')$, $\mathsf{cost}_{\vec{G}}(R_2^*) \leq \mathsf{cost}_{\vec{G}}(R_2')$, $\mathsf{cost}_{\vec{G}}(Q_2^*) \leq \mathsf{cost}_{\vec{G}}(Q_2')$ and $\mathsf{cost}_{\vec{G}}(Q_4^*) \leq \mathsf{cost}_{\vec{G}}(Q_4')$. Also by the construction, we have that $\mathsf{cost}_{\vec{G}}(R_1') + \mathsf{cost}_{\vec{G}}(R_2') = \mathsf{cost}_{\vec{G}}(Q_2') + \mathsf{cost}_{\vec{G}}(Q_4')$. Therefore, we have

$$\begin{aligned}
\mathsf{cost}_{\vec{G}}(\mathcal{S}) &= \mathsf{cost}_{\vec{G}}(\mathcal{S}_2) + \mathsf{cost}_{\vec{G}}(\mathcal{S}_4) + \mathsf{cost}_{\vec{G}}(R_1^*) + \mathsf{cost}_{\vec{G}}(R_2^*) - \mathsf{cost}_{\vec{G}}(Q_2^*) - \mathsf{cost}_{\vec{G}}(Q_4^*) \\
&\leq \mathsf{cost}_{\vec{G}}(\mathcal{S}_2) + \mathsf{cost}_{\vec{G}}(\mathcal{S}_4) + \mathsf{cost}_{\vec{G}}(R_1') + \mathsf{cost}_{\vec{G}}(R_2') - \mathsf{cost}_{\vec{G}}(Q_2^*) - \mathsf{cost}_{\vec{G}}(Q_4^*) \\
&= (\mathsf{cost}_{\vec{G}}(\mathcal{S}_2) - \mathsf{cost}_{\vec{G}}(Q_4^*)) + (\mathsf{cost}_{\vec{G}}(\mathcal{S}_4) - \mathsf{cost}_{\vec{G}}(Q_2^*)) + \mathsf{cost}_{\vec{G}}(R_1') + \mathsf{cost}_{\vec{G}}(R_2') \\
&\leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_1}) + \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_2}) \\
&= \mathsf{cost}_{\vec{G}}(\mathcal{W}_P).
\end{aligned}$$



Now suppose that D is an open walk. If $a_2 = \mathsf{nil}$ and $a_4 \in (\mathcal{A} \times \mathcal{A})$, or $a_4 = \mathsf{nil}$ and $a_2 \in (\mathcal{A} \times \mathcal{A})$, then this is immediate. If $a_2 = (u_2^*, v_2^*) \in \mathcal{A}$, $a_4 = (u_4^*, v_4^*) \in \mathcal{A}$, $l_2 = r_2$, $l_4 = r_4$ and $p_2 = p_4$, then we have $a = ((u', v'), (u'', v''))$ where $v' \in \{v_2^*, v_4^*\}$ and $u'' \in \{u_2^*, u_4^*\}$. We may assume w.l.o.g that $v' = v_2^*$ and $u'' = u_4^*$. Then we have $l = l_2$, $r = r_2$ and $p = p_2$. Let R_1 be the shortest-path in $\vec{G}$ from u' to l_2. Let R_2 be the shortest-path in $\vec{G}$ from l_1 to v'. Let R' be the path from u' to v' obtained by the concatenation of R_1 and R_2. Let Y_1 be the shortest path in $\vec{G}$ from u'' to l_4. Let Y_2 be the shortest path in $\vec{G}$ from l_4 to v''. Let Y' be the path from u'' to v'' obtained by the concatenation of Y_1 and Y_2. Let Z_1 be the shortest path in $\vec{G}$ from u_2^* to l_2. Let Z_2 be the shortest path in $\vec{G}$ from l_2 to l_4. Let Z_3 be the shortest path in $\vec{G}$ from l_4 to v_4^*. Let Z' be the path from u_2^* to v_4^* obtained by the concatenation of Z_1, Z_2 and Z_3. Let Q_2^* be the grip of $\mathcal{S}_2$ and let Q_4^* be the grip of $\mathcal{S}_4$. By the construction, we have that $\mathsf{cost}_{\vec{G}}(\mathcal{S}) = \mathsf{cost}_{\vec{G}}(\mathcal{S}_2) + \mathsf{cost}_{\vec{G}}(\mathcal{S}_4) + \mathsf{cost}_{\vec{G}}(R') + \mathsf{cost}_{\vec{G}}(Y') + \mathsf{cost}_{\vec{G}}(Z') - \mathsf{cost}_{\vec{G}}(Q_2^*) - \mathsf{cost}_{\vec{G}}(Q_4^*)$. By the construction, there exists a P-important walk from u_2^* to v_4^*, a P-important walk from u' to v_2^*, and a P-important walk from u_4^* to v''. Therefore by Proposition 63, these important walks are unique. Let R'', Y'' and Z'' be the important walks from u' to v', u'' to v'', and u_2^* to v_4^* respectively. Therefore, we have that

$$\begin{aligned}\mathsf{cost}_{\vec{G}}(\mathcal{S}) &= \mathsf{cost}_{\vec{G}}(\mathcal{S}_2) + \mathsf{cost}_{\vec{G}}(\mathcal{S}_4) + \mathsf{cost}_{\vec{G}}(R') + \mathsf{cost}_{\vec{G}}(Y') + \mathsf{cost}_{\vec{G}}(Z')\\ &\quad - \mathsf{cost}_{\vec{G}}(Q_2^*) - \mathsf{cost}_{\vec{G}}(Q_4^*)\\ &\leq \mathsf{cost}_{\vec{G}}(\mathcal{S}_2) + \mathsf{cost}_{\vec{G}}(\mathcal{S}_4) + \mathsf{cost}_{\vec{G}}(R'') + \mathsf{cost}_{\vec{G}}(Y'') + \mathsf{cost}_{\vec{G}}(Z'')\\ &\quad - \mathsf{cost}_{\vec{G}}(Q_2^*) - \mathsf{cost}_{\vec{G}}(Q_4^*)\\ &\leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_1}) + \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_2}) = \mathsf{cost}_{\vec{G}}(\mathcal{W}_P).\end{aligned}$$

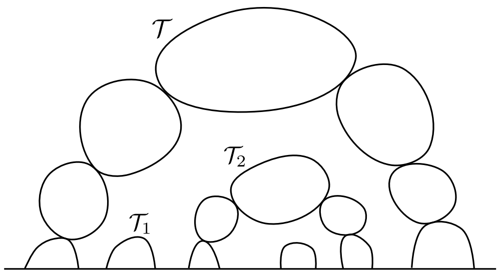
If $a_4 = ((u_4, v_4), (u_4', v_4')) \in (\mathcal{A} \times \mathcal{A})$, $a_2 = (u_2, v_2) \in \mathcal{A}$, $l_2 = r_2$ and $p_2 = p_4$, then we have $a = ((u_4, v_4), (u_2, v_4'))$, $l = l_4$, $r = r_2$ and $p = p_2$. Let (Q_4^*, Q_4^{**}) be the grip of $\mathcal{S}_4$, and let Q_2^* be the grip of $\mathcal{S}_2$. We may assume w.l.o.g that Q_4^* is a path from u_4 to v_4, and Q_4^{**} is a path from u_4' to v_4'. Let Y_1 be the shortest path in $\vec{G}$ from u_2 to l_2. Let Y_2 be the shortest path in $\vec{G}$ from l_2 to v_4'. Let Y' be the path from u_2 to v_4' obtained by the concatenation of Y_1 and Y_2. Let Z_1 be the shortest path in $\vec{G}$ from u_4' to r_4. Let Z_2 be the shortest path in $\vec{G}$ from r_4 to l_2. Let Z_3 be the shortest path in $\vec{G}$ from l_2 to v_2. Let Z' be the path from u_4' to v_2 obtained by the concatenation of Z_1, Z_2 and Z_3. By the construction, we have that $\mathsf{cost}_{\vec{G}}(\mathcal{S}) = \mathsf{cost}_{\vec{G}}(\mathcal{S}_2) + \mathsf{cost}_{\vec{G}}(\mathcal{S}_4) + \mathsf{cost}_{\vec{G}}(Y') + \mathsf{cost}_{\vec{G}}(Z') - \mathsf{cost}_{\vec{G}}(Q_2^*) - cost_{\vec{G}}(Q_4^{**})$. By the construction, there exists a P-important walk from u_4' to v_2, and a P-important walk from u_2 to v_4'. Therefore by Proposition 63, these important walks are unique. Let Y'' and Z'' be the important walks from u_2 to v_4', and u_4' to v_2 respectively. Therefore we have

$$\begin{aligned}\mathsf{cost}_{\vec{G}}(\mathcal{S}) &= \mathsf{cost}_{\vec{G}}(\mathcal{S}_2) + \mathsf{cost}_{\vec{G}}(\mathcal{S}_4) + \mathsf{cost}_{\vec{G}}(Y') + \mathsf{cost}_{\vec{G}}(Z') - \mathsf{cost}_{\vec{G}}(Q_2^*) - cost_{\vec{G}}(Q_4^{**})\\ &\leq \mathsf{cost}_{\vec{G}}(\mathcal{S}_2) + \mathsf{cost}_{\vec{G}}(\mathcal{S}_4) + \mathsf{cost}_{\vec{G}}(Y'') + \mathsf{cost}_{\vec{G}}(Z'') - \mathsf{cost}_{\vec{G}}(Q_2^*) - cost_{\vec{G}}(Q_4^{**})\\ &\leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_1}) + \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_2}) = \mathsf{cost}_{\vec{G}}(\mathcal{W}_P).\end{aligned}$$

Now suppose that $\mathcal{F}$ contains more than one tree. Let $A = \{\mathcal{T}_1, \dots, \mathcal{T}_m\}$ be the set of all trees in $\mathcal{F}$. For every $\mathcal{T} \in A$ we define the *level* of $\mathcal{T}$, $L(\mathcal{T})$, as follows. Let D be the root of $\mathcal{T}$. We set $L(\mathcal{T}) = 0$, if there exists a basic path P' that covers $\mathcal{T}$ and avoids all $\mathcal{T}' \in \mathcal{F} \setminus \{\mathcal{T}\}$. We call a minimal such path, a *corresponding* basic path for $\mathcal{T}$ and we denote it by $P_{\mathcal{T}}$; it is immediate that there is a unique such minimal path. Let $\mathcal{F}_0 = \{\mathcal{T} \in \mathcal{F} : L(\mathcal{T}) = 0\}$. Now for $i \geq 0$, suppose that we have defined trees of level i and $\mathcal{F}_i$. Suppose that $L(\mathcal{T}) \notin \{0, \dots, i\}$. We set $L(\mathcal{T}) = i + 1$ if there exists a basic path P' that covers $\mathcal{T}$ such that for all $\mathcal{T}' \in \bigcup_{j=0}^{i} \mathcal{F}_j$, P' either avoids or covers $\mathcal{T}'$, and for all $\mathcal{T}'' \in (\mathcal{F} \setminus \{\mathcal{T}\}) \setminus \bigcup_{j=0}^{i} \mathcal{F}_j$, P' avoids $\mathcal{T}''$. We also call a minimal such path corresponding basic for $\mathcal{T}$ and we denote it by $P_{\mathcal{T}}$. Let $\mathcal{F}_{i+1} = \{\mathcal{T} \in \mathcal{F} : L(\mathcal{T}) = i + 1\}$.

We say that some $\mathcal{T} \in \mathcal{F}$ is *outer-most* if there is no $\mathcal{T}' \in \mathcal{F}$ with $L(\mathcal{T}') > L(\mathcal{T})$ and $P_{\mathcal{T}} \subset P_{\mathcal{T}'}$.

Let us first suppose that there exists only one outer-most tree $\mathcal{T} \in \mathcal{F}$, such that $P_{\mathcal{T}} \subseteq P$. We will deal with the more general case later. Also, suppose that $\mathcal{T} \in \mathcal{F}_m$ for some $m \geq 0$. We will prove the assertion by induction on m. We also prove that for this case, we have $a = l = r = p = \mathsf{nil}$. For the base case, if $m = 0$, then by the construction, $\mathcal{T}$ is the only tree with a corresponding basic path $P_{\mathcal{T}} \subseteq P$. For this case, we have already established the assertion and we are done. Now suppose that we have proved the assertion for all $m' < m$. Let $\mathcal{F}' = \mathcal{F} \setminus \{\mathcal{T}\}$. Let $\mathcal{T}_1, \dots, \mathcal{T}_t \in \mathcal{F}'$ be all outer-most trees in $\mathcal{F}'$, such that for $j \in \{1, \dots, t\}$ we have $P_{\mathcal{T}_j} \subseteq P$, and they intersect F in this order. By the construction, for every $j \in \{1, \dots, t\}$ we have $L(\mathcal{T}_j) < m$. For every $j \in \{1, \dots, t\}$ let $P_{\mathcal{T}_j}$ be a corresponding basic path for $\mathcal{T}_j$. By the induction hypothesis, for every $j \in \{1, \dots, t\}$ there exists a partial solution $\mathcal{S}_j$ for $P_{\mathcal{T}_j}$. Now, we can apply the same argument when we had only one tree $\mathcal{T} \in \mathcal{F}$ for $\mathcal{T}$. Note that for each $j \in \{1, \ldots, t\}$, we have $a_j = l_j = r_j = p_j = \mathsf{nil}$. The only difference is that the intermediate basic paths here are not necessarily empty basic paths, and each $\mathcal{T}_j$ appears as an intermediate basic path, with a partial solution $\mathcal{S}_j$. Therefore, by following a similar approach to the previous cases and merging appropriately we get a partial solution $\mathcal{S}$, as desired.



Now, suppose that there exist more than one outer-most trees $\mathcal{T} \in \mathcal{F}$, where $P_{\mathcal{T}} \subseteq P$. Let $B = \{\mathcal{T}_1, \ldots, \mathcal{T}_t\}$ be the set of all such trees, where they intersect consecutive subpaths of P in this order. In this case, we prove the assertion by induction on t. For the base case where $t = 1$, we have already proved the assertion. Now suppose that we have proved the assertion for all $t' < t$. Let $B' = \{\mathcal{T}_1, \ldots, \mathcal{T}_{t-1}\}$. Let $P' \subseteq P$ be the subpath of P such that B' is the set of all outer-most trees, with $P_{\mathcal{T}_j} \subseteq P'$ for all $1 \leq j \leq t-1$. By the induction hypothesis, there exists a partial solution $\mathcal{S}'$ for P'. Let $P'' = P \subseteq P'$. By the construction, P'' is a corresponding basic path for $\mathcal{T}_t$, and thus by the induction hypothesis, there exists a partial solution S'' for P''. Therefore, by merging $\mathcal{S}'$ and $\mathcal{S}''$ we get a partial solution $\mathcal{S}$ for P, as desired. ◀

▶ **Theorem 65.** *Let $\vec{G}$ be an n-vertex $(0,0,1,p)$-nearly embeddable graph (that is, planar with a single vortex) and let $\vec{H}$ be the vortex of $\vec{G}$. Then there exists an algorithm which computes a walk $\vec{W}$ visiting all vertices in $V(\vec{H})$ of total length $\mathsf{OPT}_{\vec{G}}(V(\vec{H}))$ in time $n^{O(p)}$.*

Proof. We have $F \in \mathcal{P}$ and it is immediate to check that F is basic. Since F is a cycle, both the endpoitns of F are some vertex v°. It follows by Lemma 64 that there exists some $\phi \in \mathcal{I}_F$ such that the dynamic programming table contains a partial solution $\mathcal{S}$ at location (F, ϕ). Let $\phi = (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p)$. It follows by condition (T4) that for all $x \in B_{v^\circ}$ we have $\mathsf{in\text{-}degree}_{\mathcal{S}}(x) = \mathsf{out\text{-}degree}_{\mathcal{S}}(x)$. Thus by repeatedly merging pairs of walks that respectively terminate to and start from the same vertex, we obtain a collection $\mathcal{S}'$ of closed walks with $\mathsf{cost}_{\vec{G}}(\mathcal{S}') = \mathsf{cost}_{\vec{G}}(\mathcal{S})$ that visit all vertices in $V(\vec{H})$. By Lemma 57 we can assume that C is the trivial partition containing only one cluster, that is $C = \{\{B_{v^\circ}\}\}$. Since C is trivial, it follows by (T5) that all vertices in $V(\vec{H})$ are in the same weakly-connected component of $\bigcup_{W \in \mathcal{S}} W$. Thus all vertices in $V(\vec{H})$ are in the same strongly-connected component of $\bigcup_{W \in \mathcal{S}'} W$. Since all walks in $\mathcal{S}'$ are closed, by repeatedly shortcutting pairs of intersecting walks, we obtain a walk $\vec{W}$ with that visits all vertices in $V(\vec{H})$ with $\mathsf{cost}_{\vec{G}}(\vec{W}) = \mathsf{cost}_{\vec{G}}(\mathcal{S}') = \mathsf{cost}_{\vec{G}}(\mathcal{S}) \leq \mathsf{cost}_{\vec{G}}(\vec{W}_{\mathsf{OPT}}) = \mathsf{OPT}_{\vec{G}}$.

The running time is polynomial in the size of the dynamic programming table, which is at most $|\mathcal{P}| \cdot \max_{P \in \mathcal{P}} |\mathcal{I}_P| = O(n^2) \cdot p^{O(p)} \cdot n^{O(p)} \cdot O(n^2) = n^{O(p)}$. ◀

13 The dynamic program for traversing a vortex in a bounded genus graph

For the remainder of this section let $\vec{G}$ be a n-vertex $(0, g, 1, p)$-nearly embeddable graph. Let $\vec{H}$ be the vortex in $\vec{G}$, attached to some face $\vec{F}$. Let $\vec{G}' = \vec{G} \setminus (V(\vec{H}) \setminus (\vec{F}))$ and fix some embedding ψ of $\vec{G}'$ on a surface S of genus g. Let F be the symmetrization of $\vec{F}$. Let $\vec{W}_{\mathsf{OPT}}$ be a closed walk in $\vec{G}$ that visits all vertices in $\vec{H}$ with minimum $\mathsf{cost}_{\vec{G}}(\vec{W})$. Fix a path-decomposition $\{B_v\}_{v \in V(F)}$ of $\vec{H}$ of width p. We present a similar algorithm as in Section 12 for computing a walk traversing all vertices in $V(\vec{H})$ based on dynamic programming. By Lemma 54 we may assume w.l.o.g. that $\vec{G}$ is facially normalized and cross normalized.

13.1 The dynamic program

Let $\mathcal{Q}$ be the set of all (possible closed) subpaths of F. For any integer m, let

$$\mathcal{P}_m = \{A \subseteq \mathcal{Q} : |A| \leq m, \text{ for every } Q, Q' \in A \text{ we have } V(Q) \cap V(Q') = \emptyset\}$$

and let

$$\mathcal{P}_\infty = \{A \subseteq \mathcal{Q} : \text{ For every } Q, Q' \in A \text{ we have } V(Q) \cap V(Q') = \emptyset\}.$$

For every $P = \{Q_1, \ldots, Q_m\} \in \mathcal{P}_\infty$, let $E(P) = \bigcup_{i=1}^m E(Q_i)$ and let $V(P) = \bigcup_{i=1}^m V(Q_i)$. For each $i \in \{1, \ldots, m\}$, let u_i and v_i be the endpoints of Q_i. Similar to the planar case, let $\vec{H}_P = \vec{H}\left[\bigcup_{x \in V(P)} B_x\right]$. Let $\mathcal{B} = \bigcup_{i=1}^m (B_{u_i} \cup B_{v_i})$. Let $\mathcal{C}_P$ be the set of all possible partitions of $\mathcal{B}$. Let $\mathcal{D}_P^{\mathsf{in}} = \{0, \ldots, n\}^{\mathcal{B}}$, $\mathcal{D}_P^{\mathsf{out}} = \{0, \ldots, n\}^{\mathcal{B}}$, that is, every element of $\mathcal{D}_P^{\mathsf{in}} \cup \mathcal{D}_P^{\mathsf{out}}$ is a function $f : \mathcal{B} \to \{0, \ldots, n\}$.

13.1.1 The dynamic programming table.

Let $\mathcal{P} = \mathcal{P}_{324000g^4}$. With these definitions, the dynamic programming table is indexed the exact same way as in the planar case. Also, a *partial solution* is a collection of walks in $\vec{G}$.

We say that a partial solution $\mathcal{S}$ is *compatible* with (P, ϕ) if the same conditions (T1)-(T5) as in the planar case are satisfied. The only difference is that instead of $B_u \cup B_v$, we have $\mathcal{B}$.

13.1.1.1 Merging partial solutions

We follow a similar approach as in the planar case. Let $P = \{Q_1, \ldots, Q_m\}$, $P_1 = \{Q'_1, \ldots, Q'_{m'}\}$, $P_2 = \{Q''_1, \ldots, Q''_{m''}\} \in \mathcal{P}$ such that $E(P_1) \neq \emptyset$, $E(P_2) \neq \emptyset$, $E(P_1) \cap E(P_2) = \emptyset$, and $E(P) = E(P_1) \cup E(P_2)$. Let $\phi = (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p) \in \mathcal{I}_P$, $\phi_1 = (C_1, f_1^{\mathsf{in}}, f_1^{\mathsf{out}}, a_1, l_1, r_1, p_1) \in \mathcal{I}_{P_1}$, $\phi_2 = (C_2, f_2^{\mathsf{in}}, f_2^{\mathsf{out}}, a_2, l_2, r_2, p_2) \in \mathcal{I}_{P_2}$.

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be partial solutions compatible with (P_1, ϕ_1) and (P_2, ϕ_2) respectively. Similar to the planar case, we compute a partial solution $\mathcal{S}$ compatible with (P, ϕ) as follows.

Merging phase 1: Joining the walks. For every $w \in V(P_1) \cap V(P_2)$, we check that for all $x \in B_w$ we have $f_1^{\mathsf{in}}(x) = f_2^{\mathsf{out}}(x)$ and $f_2^{\mathsf{in}}(x) = f_1^{\mathsf{out}}(x)$. If not then the merging procedure returns nil. Otherwise, for every $w \in V(P_1) \cap V(P_2)$ and every $x \in B_w$, we follow the exact same approach as in the planar case.

Merging phase 2: Updating the grip. This phase is identical to the planar case.

Merging phase 3: Checking connectivity. Similar to the planar case, we check that condition (T5) holds for $\mathcal{S}$ and we return nil if it does not.

13.1.2 Initializing the dynamic programming table.

For all $P \in \mathcal{P}_1$ with $|E(P)| \leq 1$, we follow the same approach as in the planar case.

13.1.3 Updating the dynamic programming table.

For all $P \in \mathcal{P}$ with $|E(P)| > 1$, and for all $P_1, P_2 \in \mathcal{P}$ with $E(P_1) \neq \emptyset$, $E(P_2) \neq \emptyset$, $E(P_1) \cap E(P_2) = \emptyset$ and $E(P_1) \cup E(P_2) = E(P)$, and for all $\phi_1 \in \mathcal{I}_{P_1}$ and $\phi_2 \in \mathcal{I}_{P_2}$ we proceed as follows. Suppose that for all $P' \in \mathcal{P}$ with $|E(P')| < |E(P)|$ and all $\phi' \in \mathcal{I}_{P'}$, we have computed the partial solutions in the dynamic programming table at (P', ϕ'). Now similar to the planar case, if there exists partial solutions $\mathcal{S}_1$ and $\mathcal{S}_2$ at (P_1, ϕ_1) and (P_2, ϕ_2) respectively, we call the merging process to (possibly) get a partial solution $\mathcal{S}$ at (P, ϕ) for some $\phi \in \mathcal{I}_P$. Now similar to the planar case, if there is no partial solution at (P, ϕ) then we store $\mathcal{S}$ at (P, ϕ). Otherwise if there there exists a partial solution $\mathcal{S}'$ stored at (P, ϕ) and $\mathsf{cost}_{\vec{G}}(\mathcal{S}) < \mathsf{cost}_{\vec{G}}(\mathcal{S}')$ then we replace $\mathcal{S}'$ with $\mathcal{S}$.

13.2 Analysis

Let $\mathcal{W}$ be the collection of walks given by Lemma 56. Let $\mathcal{F}$ be the forest given by Lemma 57. Let $\mathcal{T}$ be a subtree of $\mathcal{F}$. We say that $\mathcal{T}$ is *trivial* if $\psi(F) \cup \bigcup_{D \in V(\mathcal{T})} \psi(D)$ is contractible. Otherwise, we say that $\mathcal{T}$ is *non-trivial.*

Let $Q \in \mathcal{Q}$, and let $\mathcal{T}$ be a subtree of some tree in $\mathcal{F}$. We define the terms Q *covers* $\mathcal{T}$ and Q *avoids* $\mathcal{T}$ the exact same way as in the planar case. Let $P = \{Q_1, \ldots, Q_m\} \in \mathcal{P}_\infty$. We say that P *covers* $\mathcal{T}$ if for all $D \in V(\mathcal{T})$ we have $V(D) \cap V(F) \subseteq V(Q_1 \cup \ldots \cup Q_m)$. We say that P *avoids* $\mathcal{T}$ if for all $Q_i \in P$ we have that Q_i avoids $\mathcal{T}$.

▶ **Definition 66** (Basic family of paths). Let $P = \{Q_1, \ldots, Q_m\} \in \mathcal{P}_\infty$. For each $i \in \{1, \ldots, m\}$, let u_i and v_i be the endpoints of Q_i. We say that P is *basic* (w.r.t. $\mathcal{W}$) if either $V(P) \setminus (\bigcup_{i=1}^{m} \{u_i, v_i\})$ does not intersect any of the walks in $\mathcal{W}$ (in which case we call it *empty basic*) or there exists $\mathcal{T} \in \mathcal{F}$ and $D \in V(\mathcal{T})$, with children $D_1, \ldots, D_k$, intersecting D in this order along a traversal of D, such that the exact same conditions (1) & (2) as in Definition 60 hold, and $|P|$ is minimal subject to the following:

3. If P covers $\mathcal{T}_D$ and avoids $\mathcal{T} \setminus \mathcal{T}_D$, let $\mathcal{T}[P] = \mathcal{T}_D$, and otherwise let $\mathcal{T}[P] = \bigcup_{i=1}^{j} \mathcal{T}_{D_i}$. For every two disjoint subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$ of $\mathcal{T}[P]$, the following holds. If there exists $Q_i \in P$ such that Q_i covers $\mathcal{T}_1 \cup \mathcal{T}_2$ and avoids $\mathcal{T} \setminus (\mathcal{T}_1 \cup \mathcal{T}_2)$, then there exist edge-disjoint subpaths Q_i', Q_i'' of Q_i such that $Q_i = Q_i' \cup Q_i''$, Q_i' covers $\mathcal{T}_1$ and avoids $\mathcal{T} \setminus \mathcal{T}_1$, and Q_i'' covers $\mathcal{T}_2$ and avoids $\mathcal{T} \setminus \mathcal{T}_2$ (see Figure 3 for an example).

Moreover if for each non-trivial tree $\mathcal{T}'$ in $\mathcal{F}$ with $\mathcal{T}' \neq \mathcal{T}$, we have that P avoids $\mathcal{T}'$, then we say that P is *elementary.* Furthermore, if for each non-trivial tree $\mathcal{T}'$ in $\mathcal{F}$, we have that P avoids $\mathcal{T}'$, then we say that P is *trivial.*

▶ **Definition 67** (Twins). Let $P, P' \in \mathcal{P}_\infty$. We say that P and P' are *twins* if for each subtree $\mathcal{T}$ of $\mathcal{F}$, P covers $\mathcal{T}$ if and only if P' covers $\mathcal{T}$, and P avoids $\mathcal{T}$ if and only if P' avoids $\mathcal{T}$.

▶ **Definition 68** (Succinctness). Let $P = \{Q_1, \ldots, Q_m\} \in \mathcal{P}_\infty$ be basic. For each $i \in \{1, \ldots, m\}$, let u_i and v_i be the endpoints of Q_i, let $Q_i' = Q_i \setminus \{u_i\}$ and let $Q_i'' = Q_i \setminus \{v_i\}$. We say that P is *succinct* if for all $i \in \{1, \ldots, m\}$, P and $\{Q_1, \ldots, Q_{i-1}, Q_i', Q_{i+1}, \ldots, Q_m\}$ are not twins, and moreover P and $\{Q_1, \ldots, Q_{i-1}, Q_i'', Q_{i+1}, \ldots, Q_m\}$ are not twins.

▶ **Lemma 69.** *Let $P \in \mathcal{P}_\infty$ be non-empty basic. Then there exists some succinct and basic $P' \in \mathcal{P}_\infty$ such that P and P' are twins with $E(P') \subseteq E(P)$.*

Proof. It is immediate by Definition 68. ◀

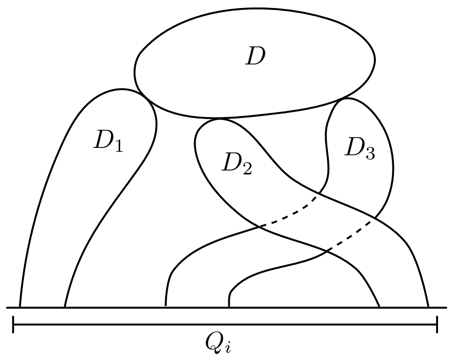


Figure 3 Example of a non-basic family of paths. $P = \{Q_i\}$ is not basic. Let $\mathcal{T}_1 = D_1 \cup D_2$ and $\mathcal{T}_2 = D_3$. Then Q_i covers $\mathcal{T}_1 \cup \mathcal{T}_2$ and avoids $\mathcal{T} \setminus (\mathcal{T}_1 \cup \mathcal{T}_2)$, but there is no edge-disjoint subpaths Q_i', Q_i'' of Q_i satisfying the third condition.

▶ **Lemma 70.** *Let $P \in \mathcal{P}_\infty$ be non-empty basic elementary and succinct. Let $\mathcal{T}$, D, j, $D_1, \ldots, D_j$ be as in Definition 66. Then there exist $P_1, P_2 \in \mathcal{P}_\infty$ satisfying the following conditions:*

1. *P_1 and P_2 are non-empty basic elementary and succinct.*
2. *P_1 covers $\bigcup_{i=1}^{j-1} \mathcal{T}_{D_i}$ and avoids $\mathcal{T} \setminus (\bigcup_{i=1}^{j-1} \mathcal{T}_{D_i})$.*
3. *P_2 covers $\mathcal{T}_{D_j}$ and avoids $\mathcal{T} \setminus \mathcal{T}_{D_j}$.*
4. *$E(P_1) \subseteq E(P)$, $E(P_2) \subseteq E(P)$, and $E(P_1) \cap E(P_2) = \emptyset$.*

Proof. We begin by defining auxiliary $Z_1, Z_2 \in \mathcal{P}_\infty$. The desired P_1 and P_2 will be succinct twins of Z_1 and Z_2. First we define Z_1. Initially, we set $Z_1 = P$ and we inductively modify Z_1 until it covers $C_1 = \bigcup_{i=1}^{j-1} \mathcal{T}_{D_i}$ and avoids $A_1 = \mathcal{T} \setminus (\bigcup_{i=1}^{j-1} \mathcal{T}_{D_i})$ as follows: If Z_1 contains a path Q that does not intersect C_1 then we remove Q from Z_1. If Z_1 contains a path Q that intersects both C_1 and A_1 then we proceed as follows: let R be the collection of paths obtained from Q by deleting all vertices in $A_1 \cap Q$; let R' be the collection obtained from R by removing all paths that do not intersect C_1; let R'' be the collection obtained from R' by replacing each $Q' \in R'$ be the minimal subpath $Q'' \subseteq Q'$ with $V(Q'') \cap C_1 = V(Q') \cap C_1$. We repeat the above process until the resulting Z_1 covers $C_1 = \bigcup_{i=1}^{j-1} \mathcal{T}_{D_i}$ and avoids $A_1 = \mathcal{T} \setminus (\bigcup_{i=1}^{j-1} \mathcal{T}_{D_i})$. In a similar fashion we define Z_2 that covers $C_2 = \mathcal{T}_{D_j}$ and avoids $A_2 = \mathcal{T} \setminus \mathcal{T}_{D_j}$. It is immediate by construction that $E(Z_1) \subset E(P)$, $E(Z_2) \subset E(P)$ and $E(Z_1) \cap E(Z_2) = \emptyset$.

Next we argue that Z_1 is basic. It is immediate that conditions (1) & (2) of Definition 66 are satisfied. It remains to establish condition (3) of Definition 66. Let $Q \in P_1$. By construction there exists $Q' \in P$ such that $Q \subseteq Q'$. Let $\mathcal{T}[Z_1]$ and $\mathcal{T}[P]$ be as in Definition 66. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be disjoint subtrees of $\mathcal{T}[Z_1]$ such that Q covers $\mathcal{T}_1 \cup \mathcal{T}_2$ and avoids $\mathcal{T} \setminus (\mathcal{T}_1 \cup \mathcal{T}_2)$. Let $\mathcal{T}_{Q'}$ be the minimal subtree of $\mathcal{T}[P]$ that contains all the nodes of $\mathcal{T}[P]$ that are covered by Q'. Let $\mathcal{T}' = \mathcal{T}_{Q'} \cup \mathcal{T}_1 \cup \mathcal{T}_2$. By definition we have that $\mathcal{T}'$ is a subtree of $\mathcal{T}[P]$. Therefore there exist disjoint subtrees $\mathcal{T}_1'$ and $\mathcal{T}_2'$ of $\mathcal{T}'$ such that $\mathcal{T}_1 \subseteq \mathcal{T}_1'$, $\mathcal{T}_2 \subseteq \mathcal{T}_2'$, and $\mathcal{T}' = \mathcal{T}_1' \cup \mathcal{T}_2'$. Since P is basic, it follows by condition (3) of Definition 66 that there exist edge-disjoint subpaths Q_1', Q_2' of Q' such that Q_1' covers $\mathcal{T}_1'$ and avoids $\mathcal{T} \setminus \mathcal{T}_1'$ and Q_2' covers $\mathcal{T}_2'$ and avoids $\mathcal{T} \setminus \mathcal{T}_2'$. Let $Q_1 = Q \cap Q_1'$ and $Q_2 = Q \cap Q_2'$. It now follows that Q_1 covers $\mathcal{T}_1$ and avoids $\mathcal{T} \setminus \mathcal{T}_1$ and Q_2 covers $\mathcal{T}_2$ and avoids $\mathcal{T} \setminus \mathcal{T}_2$, establishing Condition (3) of Definition 66.

It remains to show that $|P_1|$ is minimal subject to condition (3) of Definition 66. Suppose not. Then there are $Q, Q' \in P_1$ that are consecutive in F such that we can replace Q and Q' in P_1 by some subpath Q'' that contains Q and Q'. If Q and Q' are subpaths of the same path in P then this is a contradiction because by construction there must exist a vertex in

$V(Q'') \setminus (V(Q) \cup V(Q'))$ that P_1 must avoid. Therefore there must exist distinct $R, R' \in P$ such that $Q \subseteq R$ and $Q' \subseteq R'$. However this means that we can replace R and R' in P by some subpath containing both R and R', contradicting the fact that P is basic. This establishes that $|P_1|$ is minimal subject to condition (3) of Definition 66. We have thus obtained that Z_1 is basic. It is also immediate by construction that since P is elementary, Z_1 is also elementary. By the exact same argument it follows that Z_2 is also basic and elementary.

For any $i \in \{1, 2\}$ let P_i be a succinct twin of Z_i obtained by Lemma 69. Since Z_i is basic and elementary, it follows that P_i is also basic and elementary, and thus condition (1) is satisfied. Since P_i is a twin of Z_i, it follows that conditions (2) & (3) are satisfied. Since $E(Z_i) \subset E(P)$ and $E(P_i) \subseteq E(Z_i)$, we get $E(P_i) \subset E(P)$. Finally, since $E(Z_1) \cap E(Z_2) = \emptyset$, we have $E(P_1) \cap E(P_2) = \emptyset$, and thus condition (4) is satisfied, which concludes the proof. ◀

▶ **Definition 71** (P-facial restriction)**.** Let $P = \{Q_1, \ldots, Q_m\} \in \mathcal{P}_\infty$ be basic. We define the *P-facial restriction of $\mathcal{W}$* the exact same way as in Definition 61. We remark that P is now a family of paths, while in the planar case P is a single path; the definition remains the same by replacing the notion of basic path given in Definition 60 by the notion of basic family of paths given in Definition 66.

▶ **Lemma 72** (Malnič and Mohar [21])**.** *Let S be an either orientable or non-orientable surface of Euler genus g, and let $x \in S$. Let $\mathcal{X}$ be a collection of noncontractible curves. Suppose that at least one of the following holds:*

(i) *The curves in $\mathcal{X}$ are disjoint and (freely) nonhomotopic.*

(ii) *There exist $x \in S$ such that for every $C, C' \in \mathcal{X}$, we have $C \cap C' = x$, and the curves in $\mathcal{X}$ are nonhomotopic (in $\pi_1(S, x)$).*

Then,

$$|\mathcal{X}| \leq \begin{cases} 0 & \text{if } S \text{ is the 2-sphere} \\ 1 & \text{if } S \text{ is the torus or the projective plane} \\ 3(g-1) & \text{otherwise} \end{cases}$$

We recall the following result on the genus of the complete bipartite graph [14].

▶ **Lemma 73.** *For any $n, m \geq 1$, the Euler genus of $K_{m,n}$ is $\lceil (m-2)(n-2)/4 \rceil$.*

▶ **Lemma 74.** *Let $P \in \mathcal{P}_\infty$ be elementary and succinct. Then $|P| \leq 18000g^3$.*

Proof. Let $\mathcal{T}[P]$ be as in Definition 66. If $\mathcal{T}[P]$ is trivial, we can find $Q \in \mathcal{Q}$ such that Q covers $\mathcal{T}[P]$ and avoids $\mathcal{T} \setminus \mathcal{T}[P]$, and thus $|P| = 1$ and we are done. Now suppose that $\mathcal{T}[P]$ is non-trivial. Let $m = |P|$ and suppose that $P = \{Q_1, \ldots, Q_m\}$ such that $Q_1, \ldots, Q_m$ are subpaths of F in this order along a traversal of F. Let $Z_1, \ldots, Z_{m'}$ be a sequence of disjoint subsets of P such that $P = \bigcup_{i=1}^{m'} Z_i$, and each Z_i is maximal subject to the following condition: Let Z'_i be the minimal subpath of F that contains all the paths in Z_i and does not intersect any other paths in P; then Z'_i avoids all non-trivial tress in $\mathcal{F} \setminus \mathcal{T}$. For every $j \in \{1, \ldots, m'-1\}$, let P_j be the subpath between Z'_j and Z'_{j+1} and let $\mathcal{P}' = \{P_1, \ldots, P_{m-1}\}$. Note that since P is basic, for every two consecutive Z'_j and Z'_{j+1}, there exists a non-trivial tree $\mathcal{T}_j$ such that $\mathcal{T}_j$ intersects P_j.

We construct a set R of non-trivial trees as follows. We initially set $R = \{\mathcal{T}_1\}$. In each step $i > 1$, if there exists $\mathcal{T}' \in R$ such that $\mathcal{T}'$ intersects P_i, we continue to the next step. Otherwise, we let $\mathcal{T}'_i$ be some non-trivial tree that intersects P_i and we add $\mathcal{T}'_i$ to R and we continue to the next step. We argue that $|R| \leq 20g$. Suppose not. For each $\mathcal{T}' \in R$ where

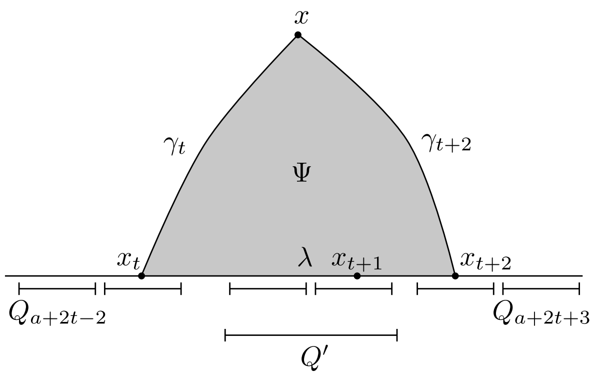
$\mathcal{T}'$ intersects some $P' \in \mathcal{P}'$ at some $x' \in V(P')$, we construct a path $\gamma_{\mathcal{T}'}$ in the surface, with both endpoints on $\psi(F)$, and such that after contracting $\psi(F)$ into single point, the loop resulting from $\gamma_{\mathcal{T}'}$ is non-contractible, as follows. First suppose that $\psi(\mathcal{T}')$ contains some non-contractible loop γ. Then let ζ be a path in $\psi(\mathcal{T}')$ between x' and some point in γ. We set $\gamma_{\mathcal{T}'}$ to be the path starting at x', traversing ζ, followed by γ, followed by the reversal of ζ, and terminating at x'. Otherwise, suppose that $\psi(cT')$ does not contain any non-contractible loops. Since $\mathcal{T}'$ is non-trivial, it follows that $\psi(\mathcal{T}')$ contains some path ξ with both endpoints in $\psi(F)$ such that after contracting $\psi(F)$ into a single point, the loop obtained from ξ is non-contractible. Let ξ' be some path in $\psi(\mathcal{T}')$ between x' and some point in ξ. By Thomassen's 3-path condition [22] it follows that $\xi \cup \xi'$ contains some path ξ'' with both endpoints in $\psi(F)$, such that one of these endpoints is x', and such that after contracting $\psi(F)$ into a single point, the loop resulting from ξ'' is non-contractible. We set $\gamma_{\mathcal{T}'}$ to be ξ''.

Let $\mathcal{L}$ be the set of all loops obtained from the paths $\gamma_{\mathcal{T}'}$ as follows. Pick a point x in the interior of the disk bounded by $\psi(F)$. Connect x to both endpoints of each $\gamma_{\mathcal{T}'}$ by paths such that all chosen paths are interior disjoint. During this process each path $\gamma_{\mathcal{T}'}$ gives rise to a non-contractible loop in $\mathcal{L}$ such that any two loops in $\mathcal{L}$ intersect only at x. Let $L', L'', L'' \in \mathcal{L}$ be distinct. We show that L', L'' and L''' can not be all homotopic. Suppose not. Since they are interior-disjoint, by removing them from the surface we obtain three connected components. Since $\psi(\mathcal{T})$ does not intersect any of L', L'' and L''', it has to be inside one of the three connected components completely. We may assume w.l.o.g that $\mathcal{T}$ is inside the component which is bounded by L' and L''. Therefore, there is no path from T to L''' without crossing $L' \cup L''$, which is a contradiction.

Let $\mathcal{L}' \subseteq \mathcal{L}$ be a maximal subset such that for all $L', L'' \in \mathcal{L}'$ we have that L' and L'' are non-homotopic. Since $|\mathcal{L}| > 20g$ and for every three loops in $\mathcal{L}'$ we know that at most two of them are homotopic, we have that $|\mathcal{L}'| > 10g$, which contradicts Lemma 72. Therefore, we have that $|R| \le 20g$ and thus there exists $\mathcal{T}_0 \in R$ such that $\mathcal{T}_0$ intersects at least $10g = 200g^2/20g$ elements of $\mathcal{P}'$. Let x_1 be a point inside the face. Let x_2 be a point in the root of $\mathcal{T}_0$ and let x_3 be a point in $\psi(D)$. There exists $P'_1, \dots, P'_{10g} \in \mathcal{P}'$ such that for every $i \in \{1, \dots, 10g\}$, $\mathcal{T}_0$ intersects P'_i. For every $i \in \{1, \dots, 10g\}$, let y_i be a point on P'_i. By the construction, for every y_i we can find non-crossing paths to x_1, x_2 and x_3. Therefore, we get an embedding of $K_{3,10g}$ in a surface of genus g which contradicts Lemma 73. This establishes that $m' \le 200g^2$.

Let $i \in \{1, \dots, m'\}$. We next we bound $|Z_i|$. Suppose $Z_i = \{Q_a, Q_{a+1}, \dots, Q_{a+\ell}\}$. Let x be an arbitrary point in $\psi(D)$. For each $j \in \{0, \dots, \lfloor(\ell-1)/2\rfloor\}$ pick an arbitrary point $x_j \in \psi(Q_{a+2j}) \cap \psi(\mathcal{T})$; we define a path in the surface between x_j and x as follows: we start from the vertex D' of $\mathcal{T}$ containing x_j; if D' is a closed walk then we traverse D' clockwise until we reach the point that connects D' to its parent; otherwise we traverse the unique path between x_j and the point that connects D' to its parent. We continue in this fashion until we reach D, and we finally traverse D clockwise until we reach x. This completes the definition of the path γ_j. It is immediate that for all $j \neq j'$, the paths γ_j and $\gamma_{j'}$ are non-crossing.

Let S' be the surface obtained by contracting $\psi(F)$ into a single point y, and identifying x with y (note that S' has Euler genus at most $g+2$). For each $j \in \{0, \dots, \lfloor(\ell-1)/2\rfloor\}$ let γ'_j be the loop in S' obtained from γ_j after the above contraction and identification. We argue that there can be at most 4 loops γ'_j that are pairwise homotopic. Suppose for the sake of contradiction that there are at least 5 such loops. It follows that there must exist $t \in \{0, \dots, \lfloor(\ell-1)/2\rfloor\}$ such that γ'_t, γ'_{t+1}, and γ'_{t+2} are all pairwise homotopic. We are going to obtain a contradiction by arguing that the paths Q_{a+2t} and Q_{a+2t+1} violate the fact that P is basic. To that end, let Q' be the minimal subpath of F that contains both Q_{a+2t} and Q_{a+2t+1} and does not intersect any other paths in P. We will show that $(P \setminus \{Q_{a+2t}, Q_{a+2t+1}\}) \cup \{Q'\}$ is basic, thus violating the fact that $|P|$ is minimal subject to condition (3) of definition 66. Let $\mathcal{T}_1$, $\mathcal{T}_2$ be disjoint subtrees of $\mathcal{T}[P]$ such that Q' covers $\mathcal{T}_1 \cup \mathcal{T}_2$ and avoids $\mathcal{T} \setminus (\mathcal{T}_1 \cup \mathcal{T}_2)$. We need to show that there exist edge-disjoint subpaths Q'_1 and Q'_2 of Q' such that $Q' = Q'_1 \cup Q'_2$, Q'_1 covers $\mathcal{T}_1$ and avoids $\mathcal{T} \setminus \mathcal{T}_1$, and Q'_2 covers $\mathcal{T}_2$ and avoids $\mathcal{T} \setminus \mathcal{T}_2$. Let λ be the subpath of $\psi(F)$ in S between x_t and x_{t+1} that contains x_{t+1}. Since γ'_t, γ'_{t+1}, and γ'_{t+2} are homotopic, it follows that $\gamma_t \cup \lambda \cup \gamma_{t+2}$ bounds a disk Ψ in S. Each x_s is contained in the image of a unique vertex D'_s of $\mathcal{T}[P]$. Let B be the path in $\mathcal{T}[P]$ between D'_t and D'_{t+2}. For each $r \in \{1, 2\}$, $\mathcal{T}_r$ intersects B into some possibly empty subpath B_r. It follows that there exist disjoint disks $\Psi_1, \Psi_2 \subset \Psi$ such that for each $r \in \{1, 2\}$, $\psi(\mathcal{T}_r) \cap \Psi \subset \Psi_r$. Therefore there exist edge-disjoint subpaths Q'_1 and Q'_2 of Q' with $Q' = Q'_1 \cup Q'_2$ such that $\psi(Q') \cap \Psi_1 \subseteq \psi(Q'_1)$ and $\psi(Q') \cap \Psi_2 \subseteq \psi(Q'_2)$. It follows that Q'_1 covers $\mathcal{T}_1$ and avoids $\mathcal{T} \setminus \mathcal{T}_1$, and Q'_2 covers $\mathcal{T}_2$ and avoids $\mathcal{T} \setminus \mathcal{T}_2$. This contradicts the fact that P is basic, and concludes the proof that there are at most four loop γ'_t that are pairwise homotopic.



Pick $I \subseteq \{0, \ldots, \lfloor(\ell-1)/2\rfloor\}$, with $|I| \geq \lfloor(\ell-1)/10\rfloor$ such that for all $t \neq t' \in I$, we have that γ'_t and $\gamma'_{t'}$ are non-homotopic. Since the paths γ_t are non-crossing, we may assume w.l.o.g. that the paths γ'_t are interior disjoint after an infinitesimal perturbation. By Lemma 72 on S' it follows that $|I| \leq 3(g+1)$. Since $|I| \geq \ell/15$, we get $\ell \leq 45(g+1)$. Therefore $|Z_i| \leq 45(g+1)$.

We conclude that $m \leq \sum_{i=1}^{m'} |Z_i| \leq m' \cdot \max_{i \in \{1,\ldots,m'\}} |Z_i| \leq 9000g^2(g+1) \leq 18000g^3$. ◀

▶ **Lemma 75.** *Let $P_1 = \{Q_1, \ldots, Q_m\} \in \mathcal{P}$ be succinct. Let $P_2 = \{Q'_1, \ldots, Q'_l\} \in \mathcal{P}$ such that P_1 and P_2 are twins, $V(P_1) \subseteq V(P_2)$ and $E(P_1) \subseteq E(P_2)$. For every $i \in \{1, \ldots, m\}$, let u_i and v_i be the endpoints of Q_i. Let $\mathcal{B}_1 = \bigcup_{i=1}^{m}(B_{u_i} \cup B_{v_i})$. Let $\mathcal{W}_{P_1}$ be the P_1-facial restriction of $\mathcal{W}$. Let $\Gamma_1 = \bigcup_{\vec{W} \in \mathcal{W}_{P_1}} \vec{W}$. Let C_1 be the partition of $\mathcal{B}_1$ that corresponds to the weakly-connected components of Γ_1. For any $x \in \mathcal{B}_1$ let $f_1^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_{P_1}}(x)$ and $f_1^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_{P_1}}(x)$. We similarly define $C_2, f_2^{\mathsf{in}}, f_2^{\mathsf{out}}$ and $\mathcal{W}_{P_2}$ for P_2. Suppose that there exists some $a_1 \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}$ and $l_1, r_1, p_1 \in (V(\vec{G}) \cup \mathsf{nil})$ such that the dynamic programming table contains some partial solution $\mathcal{S}_1$ at location $(P_1, (C_1, f_1^{\mathsf{in}}, f_1^{\mathsf{out}}, a_1, l_1, r_1, p_1))$, with $\mathsf{cost}_{\vec{G}}(\mathcal{S}_1) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_1})$. Then there exists some $a_2 \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}$ and $l_2, r_2, p_2 \in (V(\vec{G}) \cup \mathsf{nil})$ such that the dynamic programming table contains some partial solution $\mathcal{S}_2$ at location $(P_2, (C_2, f_2^{\mathsf{in}}, f_2^{\mathsf{out}}, a_2, l_2, r_2, p_2))$, with $\mathsf{cost}_{\vec{G}}(\mathcal{S}_2) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_{P_2})$.*

Proof. Let $E_1 = E(P_2) \setminus E(P_1)$. For every $e \in E_1$, let $Q_e \in \mathcal{Q}$ be the path containing a single edge e, and let $P_e = \{Q_e\}$. Note that P_e is an empty basic family of paths and

$|E(P_e)| = 1$. Therefore for every $e \in E_1$, the initialization step of the dynamic programming, finds a partial solution $\mathcal{S}_e$ for P_e. By merging $\mathcal{S}_1$ with all these partial solutions sequentially in an arbitrary order, we get a partial solution $\mathcal{S}_2$ for P_2, as desired. ◀

▶ **Lemma 76.** *Let $P = \{Q_1 \dots, Q_m\} \in \mathcal{P}$ be basic and trivial (w.r.t. $\mathcal{W}$). For every $i \in \{1, \dots, m\}$, let u_i and v_i be the endpoints of Q_i. Let $\mathcal{B} = \bigcup_{i=1}^{m}(B_{u_i} \cup B_{v_i})$. Let $\mathcal{W}_P$ be the P-facial restriction of $\mathcal{W}$. Let $\Gamma = \bigcup_{\vec{W} \in \mathcal{W}_P} \vec{W}$. Let C be the partition of $\mathcal{B}$ that corresponds to the weakly-connected components of Γ. For any $x \in \mathcal{B}$ let $f^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_P}(x)$ and $f^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_P}(x)$. Then there exists some $a \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}$ and $l, r, p \in (V(\vec{G}) \cup \mathsf{nil})$ such that the dynamic programming table contains some partial solution $\mathcal{S}$ at location $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$, with $\mathsf{cost}_{\vec{G}}(\mathcal{S}) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_P)$.*

Proof. Since P is trivial, we have that $P \in \mathcal{P}_1$, and thus the exact same argument as in Lemma 64 applies here. ◀

▶ **Lemma 77.** *Let $P = \{Q_1 \dots, Q_m\} \in \mathcal{P}$ be succinct and elementary (w.r.t. $\mathcal{W}$). For every $i \in \{1, \dots, m\}$, let u_i and v_i be the endpoints of Q_i. Let $\mathcal{B} = \bigcup_{i=1}^{m}(B_{u_i} \cup B_{v_i})$. Let $\mathcal{W}_P$ be the P-facial restriction of $\mathcal{W}$. Let $\Gamma = \bigcup_{\vec{W} \in \mathcal{W}_P} \vec{W}$. Let C be the partition of $\mathcal{B}$ that corresponds to the weakly-connected components of Γ. For any $x \in \mathcal{B}$ let $f^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_P}(x)$ and $f^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_P}(x)$. Then there exists some $a \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}$ and $l, r, p \in (V(\vec{G}) \cup \mathsf{nil})$ such that the dynamic programming table contains some partial solution $\mathcal{S}$ at location $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$, with $\mathsf{cost}_{\vec{G}}(\mathcal{S}) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_P)$.*

Proof. Let $\mathcal{T}$, D, k, $D_1, \dots, D_k$ and j be as in Definition 66. We prove the assertion by induction on $\mathcal{T}$. For the base case, where D is a leaf of $\mathcal{T}$, the same argument as in Lemma 64 applies here. Suppose that D is non-leaf. In this case, we prove the assertion by another induction on j. For the base case, where $j = 1$, the same argument as in Lemma 64 applies here. Now suppose that we have proved the assertion for all $j' < j$. Let $P_1 \in \mathcal{P}_\infty$ such that $V(P_1) \subseteq V(P)$, $E(P_1) \subseteq E(P)$, P_1 covers $\bigcup_{i=1}^{j-1} \mathcal{T}_{D_i}$ and avoids $\mathcal{T} \setminus (\bigcup_{i=1}^{j-1} \mathcal{T}_{D_i})$. Let $P_2 \in \mathcal{P}_\infty$ such that $E(P_2) = E(P) \setminus E(P_1)$. By the construction, P_1 and P_2 are elementary. Therefore, by Lemma 69, there exists elementary and succinct $P_1' \in \mathcal{P}_\infty$ and $P_2' \in \mathcal{P}_\infty$ for P_1 and P_2 respectively. Moreover, by Lemma 70, we have that $E(P_1') \subseteq E(P)$, $E(P_2') \subseteq E(P)$ and $E(P_1') \cap E(P_2') = \emptyset$. By Lemma 74 we have that $P, P_1', P_2' \in \mathcal{P}_{18000g^3}$. We next define some $P_1'' \in \mathcal{P}_\infty$. Let Γ be the graph obtained as the union of all paths in P, that is $\Gamma = \bigcup_{Q \in P} Q$. Similarly, let $\Gamma' = \bigcup_{Q \in P_2'} Q$, let $\Gamma'' = \Gamma \setminus E(\Gamma')$, and let Γ''' be the graph obtained from Γ'' by deleting all isolated vertices. We define P_1'' to be the set of connected components in Γ'''. Note that Γ''' has at most $36000g^3$ connected components, and each such component is a path. Thus $P_1'' \in \mathcal{P}_{36000g^3}$, and $E(P_1') \subseteq E(P_1'')$. By the induction hypothesis, there exists a partial solution $\mathcal{S}_1'$ for P_1', and thus by Lemma 75, there exists a partial solution $\mathcal{S}_1''$ for P_1''. Also, by the induction hypothesis, there exists a partial solution $\mathcal{S}_2'$ for P_2'. By merging $\mathcal{S}_1''$ and $\mathcal{S}_2'$, we get a partial solution $\mathcal{S}$ for P, as desired. ◀

We define the following three types of non-trivial trees in $\mathcal{F}$:

1. We say that a non-trivial tree $\mathcal{T}$ is of the *first type*, if there exists a non-leaf $D \in V(\mathcal{T})$, such that D is a closed walk, with $V(D) \cap V(F) = \emptyset$, such that $\psi(D)$ is non-contractible.
2. We say that a non-trivial tree $\mathcal{T}$ is of the *second type*, if it is not of the first type and there exists a leaf $D \in V(\mathcal{T})$ such that $\psi(D) \cup \psi(F)$ is non-contractible.
3. We say that a non-trivial tree $\mathcal{T}$ is of the *third type*, if it is not of the first type nor of the second type.

We say that a non-trivial tree $\mathcal{T}$ is *good*, if at least one of the following conditions holds:

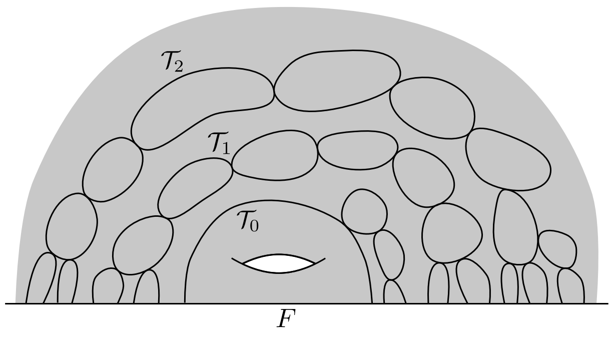


Figure 4 Example of good non-trivial trees $\mathcal{T}_0$, $\mathcal{T}_1$, and $\mathcal{T}_2$ that are pairwise friends; note that $\mathcal{T}_0$ is of the second type while $\mathcal{T}_1$ and $\mathcal{T}_2$ are of the third type.

1. $\mathcal{T}$ is of the second type, and for every $D_1, D_2 \in V(\mathcal{T})$ where $\psi(D_1) \cup \psi(F)$ and $\psi(D_1) \cup \psi(F)$ are non-contractible, we have that the loops obtained from $\psi(D_1)$ and $\psi(D_1)$ by contracting $\psi(F)$ into a single poit x are homotopic in $\pi_1(S, x)$. Let $D \in V(\mathcal{T})$ such that $\psi(D) \cup \psi(F)$ is non-contractible. We let $\beta(\mathcal{T})$ to be the homotopy class of the loop $\psi(D)$ in the surface obtained after contracting $\psi(F)$ into a single point.
2. $\mathcal{T}$ is of the third type and the following holds. Let $X = \psi(F) \cup \bigcup_{D \in V(\mathcal{T})} \psi(D)$ and let X' be the image of X after contracting $\psi(F)$ into a single point x. Then and all non-contractible loops in X' are homotopic in $\pi_1(S, x)$. We let $\beta(\mathcal{T})$ be the homotopy class in $\pi_1(S, x)$ of all non-contractible loops in X'; note that we may always take a non-contractible loop in X' that contains the basepoint x since X is connected.

Otherwise, we say that $\mathcal{T}$ is a *bad* tree.

Let $\mathcal{T}_0$ and $\mathcal{T}_1$ be non-trivial good trees in $\mathcal{F}$. We say that $\mathcal{T}_0$ is a *friend* of $\mathcal{T}_1$ if $\beta(\mathcal{T}_0) = \beta(\mathcal{T}_1)$ (see Figure 4 for an example).

▶ **Definition 78** (Friendly). Let $P \in \mathcal{P}_\infty$. We say that P is *friendly* if the following holds.
1. P avoids all non-trivial bad trees.
2. For any two non-trivial good trees $\mathcal{T}_0$ and $\mathcal{T}_1$ in $\mathcal{F}$ that P covers, we have that $\beta(\mathcal{T}_0) = \beta(\mathcal{T}_1)$.
3. If P covers a non-trivial good tree $\mathcal{T}_0$, then P covers all non-trivial good trees $\mathcal{T}_1$ with $\beta(\mathcal{T}_0) = \beta(\mathcal{T}_1)$.

▶ **Lemma 79.** *There exists at most $12g$ bad trees in $\mathcal{F}$.*

Proof. We partition all bad trees in $\mathcal{F}$ into three sets:
1. $F_1 = \{\mathcal{T} \in \mathcal{F} : \mathcal{T}$ is a bad tree of the first type$\}$.
2. $F_2 = \{\mathcal{T} \in \mathcal{F} : \mathcal{T}$ is a bad tree of the second type$\}$.
3. $F_3 = \{\mathcal{T} \in \mathcal{F} : \mathcal{T}$ is a bad tree of the third type$\}$.

We first bound $|F_1|$. Let $\mathcal{T} \in F_1$. We let $\beta(\mathcal{T})$ to be the homotopy class of some non-leaf $D \in V(T)$, such that D is a closed walk, with $V(D) \cap V(F) = \emptyset$, and $\psi(D)$ is non-contractible. Note that by the definition of trees of the first type, such a non-leaf $D \in V(\mathcal{T})$ exists. Let $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3 \in F_1$ be distinct. We show that $\beta(\mathcal{T}_1) = \beta(\mathcal{T}_2) = \beta(\mathcal{T}_3)$ cannot happen. Suppose not, and we have that $\beta(\mathcal{T}_1) = \beta(\mathcal{T}_2) = \beta(\mathcal{T}_3)$. For any $i \in \{1, 2, 3\}$, let $D_i \in V(\mathcal{T}_i)$ such that $\beta(\mathcal{T}_i)$ is the homotopy class of D_i. By removing $\psi(D_1)$, $\psi(D_2)$ and $\psi(D_3)$ from the surface we obtain three connected components. We may assume w.l.o.g that $\psi(D_2)$ is inside an annulus bounded by $\psi(D_1)$ and $\psi(D_3)$. Therefore, there is no path in the surface from

$\psi(D_2)$ to $\psi(F)$, that does not intersect $\psi(D_1 \cup D_3)$, which is a contradiction. Therefore by Lemma 72 we have that $|F_1| \leq 6g$.

Next we bound $|F_2|$ and $|F_3|$. Let $\mathcal{T} \in F_2$. By the construction, there exist $D_1, D_2 \in V(\mathcal{T})$ where $\psi(D_1) \cup \psi(F)$ and $\psi(D_2) \cup \psi(F)$ are non-contractible, and the loops obtained from $\psi(D_1)$ and $\psi(D_2)$ after contracting $\psi(F)$ into a single point y are non-homotopic in $\pi_1(S, y)$. Let x be a point in the interior of the disk bounded by $\psi(F)$. For every $\mathcal{T} \in F_2$, similarly to Lemma 74, we construct two non-homotopic loops $\gamma_{\mathcal{T}'}$ and $\gamma'_{\mathcal{T}'}$ in the surface, corresponding to D_1 and D_2 respectively, such that they only intersect at x. Let $\mathcal{L}$ be the set of all these loops. Let $L', L'', L''' \in \mathcal{L}$ be distinct. Similarly to Lemma 74, we show that L', L'' and L''' can not be all homotopic. Suppose not. We may assume w.l.o.g. that L'' is inside the disk bounded by L' and L'''. Let $\mathcal{T}'' \in F_2$ be the tree corresponding to L''. Now note that L'' is inside the disk bounded by L' and L''', and thus all non-contractible loops in $\psi(F) \cup \bigcup_{D \in V(\mathcal{T})} \psi(D)$ are in the same homotopy class as L''. This contradicts the fact that $\psi(D_1) \cup \psi(F)$ and $\psi(D_2) \cup \psi(F)$ are non-homotopic. Therefore, by Lemma 72 we have that $|F_2| \leq 3g$. Using a similar argument, we can show that $|F_3| \leq 3g$.

Therefore, we have that $|F_1| + |F_2| + |F_3| \leq 12g$, completing the proof. ◀

▶ **Lemma 80.** *Let $P = \{Q_1 \ldots, Q_m\} \in \mathcal{P}_\infty$ be friendly, succinct and basic (w.r.t. $\mathcal{W}$). For every $i \in \{1, \ldots, m\}$, let u_i and v_i be the endpoints of Q_i. Let $\mathcal{B} = \bigcup_{i=1}^{m} (B_{u_i} \cup B_{v_i})$. Let $\mathcal{W}_P$ be the P-facial restriction of $\mathcal{W}$. Let $\Gamma = \bigcup_{\vec{W} \in \mathcal{W}_P} \vec{W}$. Let C be the partition of $\mathcal{B}$ that corresponds to the weakly-connected components of Γ. For any $x \in \mathcal{B}$ let $f^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_P}(x)$ and $f^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_P}(x)$. Then there exists some $a \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}$ and $l, r, p \in (V(\vec{G}) \cup \mathsf{nil})$ such that the dynamic programming table contains some partial solution $\mathcal{S}$ at location $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$, with $\mathsf{cost}_{\vec{G}}(\mathcal{S}) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_P)$.*

Proof. If P does not cover any non-trivial trees in $\mathcal{F}$, then P is trivial and thus we have that $P \in \mathcal{P}_1$ and by Lemma 76 there exists a partial solution $\mathcal{S}$ for P. Otherwise suppose that P covers a non-trivial good tree $\mathcal{T}$ in $\mathcal{F}$. By the definition of friendly, for every non-trivial good tree $\mathcal{T}'$ in $\mathcal{F}$ with $\beta(\mathcal{T}) = \beta(\mathcal{T}')$, we have that P covers $\mathcal{T}'$. Let $A = \{\mathcal{T}' \in \mathcal{F} : \beta(\mathcal{T}) = \beta(\mathcal{T}')\}$. Suppose that $A = \{\mathcal{T}_0, \ldots, \mathcal{T}_m\}$ such that they intersect F in this order along a traversal of F. For any $i \in \{0, \ldots, m\}$, let $P_i \in \mathcal{P}_\infty$ be elementary and succinct such that P_i covers $\mathcal{T}_i$ and avoids any other non-trivial trees in $\mathcal{F}$. By Lemma 74 we have that $P_i \in \mathcal{P}_{18000g^3}$. Moreover, by Lemma 77 we have that there exists a partial solution $\mathcal{S}_i$ for P_i. For any $i \in \{0, \ldots, m\}$, let $P'_i \in \mathcal{P}_\infty$ be succinct and basic, such that P'_i covers $\bigcup_{j=0}^{i} \mathcal{T}_j$ and avoids any other non-trivial trees in $\mathcal{F}$. By the construction, each P'_i can be obtained by merging P'_{i-1} with P_i and some trivial and empty basic family of paths, and thus we have that $P'_i \in \mathcal{P}_{36000g^3}$. Therefore, by induction and Lemma 76, there exists a partial solution $\mathcal{S}'_i$ for each P'_i, and thus there exists a partial for solution $\mathcal{S}'_m$ for P'_m. Note that $P'_m = P$. Therefore, we get a partial solution $\mathcal{S}$ for P, as desired. ◀

▶ **Lemma 81.** *Let $P = \{F\}$. Let $v^\circ \in V(F)$. Let $\mathcal{W}_P$ be the P-facial restriction of $\mathcal{W}$. Let $\Gamma = \bigcup_{\vec{W} \in \mathcal{W}_P} \vec{W}$. Let C be the partition of B_{v° that corresponds to the weakly-connected components of Γ. For any $x \in B_{v^\circ}$ let $f^{\mathsf{in}}(x) = \mathsf{in\text{-}degree}_{\mathcal{W}_P}(x)$ and $f^{\mathsf{out}}(x) = \mathsf{out\text{-}degree}_{\mathcal{W}_P}(x)$. Then there exists some $a \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{A}) \cup \mathsf{nil}$ and $l, r, p \in (V(\vec{G}) \cup \mathsf{nil})$ such that the dynamic programming table contains some partial solution $\mathcal{S}$ at location $(P, (C, f^{\mathsf{in}}, f^{\mathsf{out}}, a, l, r, p))$, with $\mathsf{cost}_{\vec{G}}(\mathcal{S}) \leq \mathsf{cost}_{\vec{G}}(\mathcal{W}_P)$.*

Proof. We first partition F to the sets of basic families of paths as follows:

1. $F_1 = \{P' \in \mathcal{P}_\infty : P'$ is elementary and succinct, P' covers some bad tree $\mathcal{T}\}$.
2. $F_2 = \{P' \in \mathcal{P}_\infty : P'$ is basic, succinct and friendly$\}$.

3. $F_3 = \{P' \in \mathcal{P}_\infty : P'$ is basic, succinct and trivial$\}$.
4. $F_4 = \{P' \in \mathcal{P}_\infty : P'$ is basic, succinct and empty$\}$.

For every bad tree $\mathcal{T}$ in $\mathcal{F}$, let $P_1 \in F_1$ such that P_1 covers $\mathcal{T}$ and avoids any other non-trivial tree in $\mathcal{F}$. Since P_1 is elementary and succinct, by Lemma 74 we have that $|P_1| \leq 18000g^3$. Moreover by Lemma 77 there exists a partial solution $\mathcal{S}_1$ for P_1. Also by Lemma 79, there exist at most $12g$ bad trees in $\mathcal{F}$, and thus $|F_1| \leq 12g$. Therefore there exists $P'_1 \in \mathcal{P}_{216000g^4}$, such that for any $\mathcal{T}' \in \mathcal{F}$, P'_1 covers (resp. avoids) $\mathcal{T}'$ if and only if there exists $P_1 \in F_1$ such that P_1 covers (resp. avoids) $\mathcal{T}'$. Moreover the dynamic programming table contains a partial solution $\mathcal{S}'_1$ for P'_1. Now we will show that $|F_2| \leq 3g$. For every $P_2 \in F_2$, let $\mathcal{T}_2$ be some non-trivial good tree that P_2 covers. Let x be a point in the interior of the disk bounded by $\psi(F)$. Similar to Lemma 74 we construct a non-contractible loop γ_{P_2} that contains x, corresponding to $\mathcal{T}_2$. Let L be the set of all these interior-disjoint loops. For every $P', P'' \in F_2$, since P' and P'' are succinct and friendly, $\gamma_{P'}$ and $\gamma_{P''}$ are not homotopic. Therefore by Lemma 72 we have that $|L| \leq 3g$, and thus $|F_2| \leq 3g$. Furthermore for every $P_2 \in F_2$, by Lemma 80, we have that $P_2 \in \mathcal{P}_{36000g^3}$, and there exists a partial solution $\mathcal{S}_2$ for P_2. Therefore there exists $P'_2 \in \mathcal{P}_{108000g^4}$, such that for any $\mathcal{T}' \in \mathcal{F}$, P'_2 covers (resp. avoids) $\mathcal{T}'$ if and only if there exists $P_2 \in F_2$ such that P_2 covers (resp. avoids) $\mathcal{T}'$. Moreover the dynamic programming table contains a partial solution $\mathcal{S}'_2$ for P'_2. Let $P'_{1,2} \in \mathcal{P}_{324000g^4}$ such that for any $\mathcal{T}' \in \mathcal{F}$, $P'_{1,2}$ covers (resp. avoids) $\mathcal{T}'$ if and only if there exists $P' \in F_1 \cup F_2$ such that P' covers (resp. avoids) $\mathcal{T}'$. By merging $\mathcal{S}'_1$ and $\mathcal{S}'_2$ we get a partial solution $\mathcal{S}'_{1,2}$ for $P'_{1,2}$.

Let

$$F_5 = \{\{Q\} \in F_3 \cup F_4 : Q \text{ is maximal subject to } E(Q) \cap E(P'_{1,2}) = \emptyset\}.$$

For every $P_3 \in F_3$, by Lemma 76 we have that $P_3 \in \mathcal{P}_1$ and there exists a partial solution $\mathcal{S}_3$ for P_3. Finally, for every $P_4 \in F_4$, we have that $P_4 \in \mathcal{P}_1$ and also the dynamic programming table contains a partial solution $\mathcal{S}_4$ for P_4. Therefore for every $\{Q\} \in F_5$, the dynamic programming table contains a partial solution for $\{Q\}$. By merging these partial solutions with $P'_{1,2}$ in an arbitrary order, we get a partial solution $\mathcal{S}$, as desired. We remark that after merging the current partial solution with the partial solution for some $\{Q\} \in F_5$, we obtain a new partial solution for some $P \in P_\infty$ with fewer paths. Therefore for all intermediate $P \in P_\infty$ we have $P \in P_{324000g^4}$, concluding the proof. ◀

▶ **Theorem 82.** *Let $\vec{G}$ be an n-vertex $(0, g, 1, p)$-nearly embeddable graph and let $\vec{H}$ be the vortex of $\vec{G}$. Then there exists an algorithm which computes a walk $\vec{W}$ visiting all vertices in $V(\vec{H})$ of total length $\mathsf{OPT}_{\vec{G}}(V(\vec{H}))$ in time $n^{O(pg^4)}$.*

Proof. Let $P = \{F\}$. By Lemma 81, there exists a partial solution for P. Now the exact same argument as in Theorem 65 applies here. ◀

14 The algorithm for traversing a vortex in a nearly-embeddable graph

In this Section we obtain an exact algorithm for computing a closed walk that traverses all the vertices in the single vortex of a nearly-embeddable graph.

Proof of Theorem 6. The algorithm is as follows. Let $A \subset V(\vec{G})$ be the set of apices and let $\vec{F}$ be the face on which the vortex is attached. For each $A' \subseteq A$ we construct a $(0, g, 1, p+a)$-nearly embeddable graph $\vec{G}_{A'}$ as follows: Initially we set $\vec{G}_{A'} = \vec{G} \setminus A$. We then add A' to

$V(\vec{G}_{A'})$ and for every $u \in A'$ and every $v \in V(\vec{F})$ we add edges (u,v) and (v,u) of length $d_{\vec{G}}(u,v)$ and $d_{\vec{G}}(v,u)$ respectively; let $E_{A'}$ be the set of all these new edges. We consider A' as being part of the vortex and we modify the path decomposition of the vortex by adding A' to each one of its bubbles; it is immediate that the result is a path decomposition of width at most $a+p$. We then run the algorithm of Theorem 82 on $\vec{G}_{A'}$ and find an optimal closed walk visiting all vertices in $V(\vec{H}) \cup A'$. Let $\vec{W}_{A'}$ be the resulting walk in $\vec{G}_{A'}$. We obtain a walk $\vec{W}'_{A'}$ in $\vec{G}$ visiting all vertices in $V(\vec{H})$ with $\mathsf{cost}_{\vec{G}}(\vec{W}'_{A'}) = \mathsf{cost}_{\vec{G}_{A'}}(\vec{W}_{A'})$ by replacing every edge in $(u,v) \in E_{A'}$ by a shortest path from u to v in $\vec{G}$. After considering all subsets $A' \subseteq A$, we output the walk $\vec{W}'_{A'}$ of minimum cost that we find. This completes the description of the algorithm.

It is immediate that the running time is $O(2^a n^{(a+p)g^4}) = O(n^{(a+p)g^4})$. It remains to show that the algorithm computes an optimum walk. Let $\vec{R}$ be the walk computed by the algorithm. Let $\vec{W}_{\mathsf{OPT}}$ be a walk of minimum cost in $\vec{G}$ visiting all vertices in $V(\vec{H})$. Let A^* be the set of apices visited by $\vec{W}_{\mathsf{OPT}}$, that is $A^* = A \cap V(\vec{W}_{\mathsf{OPT}})$. Let $\vec{G}'$ be the genus-g piece of $\vec{G}$. We can construct a walk $\vec{Z}$ in $\vec{G}_{A^*}$ that visits all the vertices in $V(\vec{H}) \cup A^*$ of cost $\mathsf{cost}_{\vec{G}}(\vec{W}_{\mathsf{OPT}})$ as follows: we replace every sub-walk of $\vec{W}_{\mathsf{OPT}}$ that is contained in $\vec{G}'$, has endpoints $u, v \in V(\vec{F})$, and visits some apex $w \in A^*$, by the path u, w, v in $\vec{G}_{A^*}$. It is immediate that the resulting walk does not traverse any apices and therefore it is contained in $\vec{G}_{A^*}$. Thus we have $\mathsf{cost}_{\vec{G}}(\vec{R}) \le \mathsf{cost}_{\vec{G}}(\vec{W}'_{A^*}) = \mathsf{cost}_{\vec{G}_{A^*}}(\vec{W}_{A^*}) \le \mathsf{cost}_{\vec{G}}(\vec{W}_{\mathsf{OPT}}) \le \mathsf{OPT}_{\vec{G}}$, which concludes the proof. ◀

15 The lower bound for graphs of bounded pathwidth

In this section we present the proof of Theorem 2. This is done via the following chain of reductions:

$$\begin{tikzcd}
\boxed{\text{Clique}} \arrow[r, Rightarrow, "\text{folklore}"] & \boxed{\text{Multicolored Biclique}} \arrow[r, Rightarrow, "\text{Lemma 86}"] & \boxed{\text{Edge Balancing}} \arrow[r, Rightarrow, "\text{Lemma 90}"] & \boxed{\text{Constrained Closed Walk}} \arrow[r, Rightarrow, "\text{Lemma 87}"] & \boxed{\text{ATSP}}
\end{tikzcd}$$

15.1 Edge Balancing

Let D be a directed graph and let $\chi : E(D) \to \mathbb{Z}^+$ be an assignment of integers to the edges. We can extend χ to a set $F \subseteq E(D)$ of edges in the obvious way by defining $\chi(F) = \sum_{e \in F} \chi(e)$. Let $\delta^+_D(v)$ and $\delta^-_D(v)$ be the set of outgoing and incoming edges of v, respectively. We say that χ is *balanced* at $v \in V(D)$ if $\sum_{e \in \delta^+_D(v)} \chi(e) = \sum_{e \in \delta^-_D(v)} \chi(e)$ holds and we say that χ is *balanced* if it is balanced at every $v \in V(D)$.

EDGE BALANCING: Given a directed graph D and a set X_e of positive integers for every edge $e \in E(D)$, the task is to select a $\chi(e) \in X_e$ for every edge $e \in E(D)$ such that χ is balanced.

An easy counting argument shows that it is sufficient to require that χ balanced at all but one vertex:

▶ **Proposition 83.** *If χ is balanced at every vertex of $V(D) \setminus \{v\}$, then it is also balanced at v.*

We give a lower bound for EDGE BALANCING by a parameterized reduction from MULTICOLORED BICLIQUE.

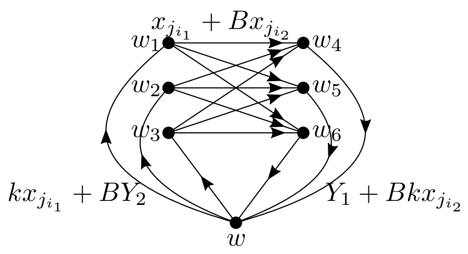


Figure 5 The instance of Edge Balancing constructed in the proof of Lemma 86. The values on the edges indicate the value of χ corresponding to a solution $(v_{i,j_i})_{i=1,\ldots,2k}$ of the Multicolored Biclique instance (we have $Y_1 = \sum_{1\le i\le k} x_{j_i}$ and $Y_2 = \sum_{k+1\le i\le 2k} x_{j_i}$).

> Multicolored Biclique: Given a graph G with a partition $(V_1, \ldots, V_{2k})$ of vertices, find a vertex $v_i \in V_i$ for every $1 \le i \le 2k$ such that v_{i_1} and v_{i_2} are adjacent for every $1 \le i_1 \le k$ and $k+1 \le i_2 \le 2k$.

There is a very simple folklore reduction from Clique to Multicolored Biclique (see, e.g., [6]) where the output parameter equals the input one, hence the lower bound of Chen et al. [5] for Clique can be transferred to Multicolored Biclique.

▶ **Theorem 84.** *Assuming ETH,* Multicolored Biclique *cannot be solved in time $f(k)n^{o(k)}$ for any computable function f.*

The reduction from Multicolored Biclique to Edge Balancing uses the technique of k-non-averaging sets to encode vertices [10, 17]. We say that a set X of positive integers is *k-non-averaging* if for every choice of k (not necessarily distinct) integers $x_1, \ldots, x_k \in X$, their average is in X only if $x_1 = x_2 = \cdots = x_k$. For example, $X = \{(k+1)^i \mid 1 \le i \le n\}$ is certainly a k-non-averaging set of size n. However, somewhat surprisingly, it is possible to construct much denser k-non-averaging sets where each integer is polynomially bounded in k and the size n of the set.

▶ **Lemma 85** (Jansen *et al.* [17]). *For every k and n there exists a k-non-averaging set X of n positive integers such that the largest element of X has value at most $32k^2n^2$. Furthermore, X can be constructed in time $O(k^2n^3)$ time.*

▶ **Lemma 86.** *Assuming ETH,* Edge Balancing *has no $f(k)n^{o(k)}$ time algorithm for any computable function f, where k is the number of vertices of D.*

Proof. The proof is by reduction from Multicolored Biclique. Let G be an undirected graph with a partition $V_1, \ldots, V_{2k}$ of the vertices in $V(G)$. By padding the instance with isolated vertices, we may assume without loss of generality that each V_i has the same number n of vertices; let $v_{i,j}$ be the j-th vertex of V_i. We construct an instance of Edge Balancing on a directed graph D having $2k+1$ vertices $w, w_1, w_2, \ldots, w_{2k}$. Let us use Lemma 85 to construct a k-non-averaging set $X = \{x_1, \ldots, x_n\}$ of n positive integers such that the maximum value in X is $M = O(k^2n^2)$. Let $B = 2kM$.

The edges of D and the sets of integers on them are constructed the following way (see Figure 5). For every $1 \le i_1 \le k$ and $k+1 \le i_2 \le 2k$, we introduce the edge (w_{i_1}, w_{i_2}) into D and we define the set $X_{(w_{i_1}, w_{i_2})}$ the following way: for every edge $v_{i_1,j_1}v_{i_2,j_2}$ between V_{i_1} and V_{i_2}, let us introduce the positive integer $x_{j_1} + Bx_{j_2}$ into $X_{(w_{i_1}, w_{i_2})}$. Next, for every $1 \le i \le k$, we introduce the edge (w, w_i) and let $X_{(w,w_i)} = \{kx + By \mid x \in X, 1 \le y \le kM\}$. Finally, for every $k+1 \le i \le 2k$, we introduce the edge (w_i, w) and let $X_{(w_i,w)} = \{y + Bkx \mid x \in X, 1 \le y \le kM\}$. This completes the description of the reduction.

Biclique ⇒ balanced assignment χ. Suppose that $v_{i,j_i} \in V_i$ for $1 \leq i \leq 2k$ form a solution of Multicolored Biclique. We define $\chi : E(D) \to \mathbb{Z}^+$ the following way. For every $1 \leq i_1 \leq k$ and $k+1 \leq i_2 \leq 2k$, we set (see the values on the edges in Figure 5)

- $\chi((w_{i_1}, w_{i_2})) = x_{j_{i_1}} + Bx_{j_{i_2}}$,
- $\chi((w, w_{i_1})) = kx_{j_{i_1}} + BY_2$, where $Y_2 = \sum_{k+1 \leq i \leq 2k} x_{j_i}$, and
- $\chi((w_{i_1}, w)) = Y_1 + Bkx_{j_{i_2}}$, where $Y_1 = \sum_{1 \leq i \leq k} x_{j_i}$.

Note that $\chi(e) \in X_e$ holds for every edge $e \in E(D)$. Let us verify that χ is balanced. For any $1 \leq i_1 \leq k$, we have $\chi(\delta^+_D(w_{i_1})) = \sum_{k+1 \leq i_2 \leq 2k}(x_{j_{i_1}} + Bx_{j_{i_2}}) = kx_{j_{i_1}} + BY_2 = \chi((w, w_{i_1})) = \chi(\delta^-_D(w_{i_1}))$, as required. Similarly, for for any $k+1 \leq i_2 \leq 2k$, we have $\chi(\delta^-_D(w_{i_2})) = \sum_{1 \leq i_1 \leq k}(x_{j_{i_1}} + Bx_{j_{i_2}}) = Y_1 + Bkx_{j_{i_2}} = \chi((w_{i_2}, w)) = \chi(\delta^+_D(w_{i_2}))$. Thus we have shown that χ is balanced at $w_1, \ldots, w_{2k}$ and it follows by Proposition 83 that χ is balanced also at w.

Balanced assignment χ ⇒ biclique. For the reverse direction of the equivalence, suppose that $\chi : E(D) \to \mathbb{Z}^+$ is a balanced assignment with $\chi(e) \in X_e$ for every $e \in E(D)$. For every $1 \leq i_1 \leq k$, the definition of $X_{(w,w_{i_1})}$ implies that $\chi((w, w_{i_1}))$ is of the form $kx_{j_{i_1}} + By_{i_1}$ where $x_{j_{i_1}} \in X$ for some $1 \leq j_{i_1} \leq n$ and y_{i_1} is a positive integer. As $kx_{j_{i_1}} \leq kM < B$ follows from $x_{j_{i_1}} \in X$, the value of $\chi((w, w_{i_1}))$ uniquely determines j_{i_1} and y_{i_1}. Similarly, for every $k+1 \leq i_2 \leq 2k$, we have that $\chi((w_{i_2}, w))$ is of the form $y_{i_2} + Bkx_{j_{i_2}}$ for uniquely determined positive integers $1 \leq j_{i_2} \leq n$ and y_{i_2}.

Having defined the values $j_1, \ldots, j_{2k}$, we show that $\chi((w_{i_1}, w_{i_2})) = x_{j_{i_1}} + Bx_{j_{i_2}}$ for every $1 \leq i_1 \leq k$ and $k+1 \leq i_2 \leq 2k$. If this is true, then the vertices $v_{i,j_i} \in V_i$ for $1 \leq i \leq 2k$ form a solution of Multicolored Biclique: the fact that $x_{j_{i_1}} + Bx_{j_{i_2}}$ was introduced into $X_{(w_{i_1},w_{i_2})}$ implies that there is an edge between $v_{i_1,j_{i_1}} \in V_{i_1}$ and $v_{i_2,j_{i_2}} \in V_{i_2}$.

As the balance requirement holds at vertex w_{i_1}, it also holds if we count modulo B. We have that $\chi((w, w_{i_1}))$ modulo B is exactly $kx_{j_{i_1}} < B$ (here we use that $kM < B$). For every $k+1 \leq i_2 \leq 2k$, the value $\chi((w_{i_1}, w_{i_2}))$ modulo B is an integer from X and the value $\chi(\delta^+_D(w_{i_1}))$ modulo B is exactly the sum of these k integers from X (as again by $kM < B$, this sum cannot reach B). Therefore, we have that the sum of k integers from X is exactly $kx_{j_{i_1}}$. Since X is a k-non-averaging set, this is only possible if these k integers are all equal to $x_{j_{i_1}}$. Thus we have shown that $\chi((w_{i_1}, w_{i_2})) = x_{j_{i_1}}$ modulo B for every $1 \leq i_1 \leq k$ and $k+1 \leq i_2 \leq 2k$.

Let us consider now a vertex w_{i_2} for $k+1 \leq i_2 \leq 2k$. The balance requirement in particular implies that $\lfloor \chi(\delta^+_D(w_{i_2}))/B \rfloor = \lfloor \chi(\delta^-_D(w_{i_2}))/B \rfloor$. First, we have $\lfloor \chi(\delta^+_D(w_{i_2}))/B \rfloor = \lfloor \chi((w_{i_2}, w))/B \rfloor = kx_{j_{i_2}}$. Therefore, the fact that χ is balanced at w_{i_2} implies $kx_{j_{i_2}} = \lfloor \chi(\delta^-_D(w_{i_2}))/B \rfloor = \left\lfloor \sum_{1 \leq i_1 \leq k} \chi((w_{i_1}, w_{i_2}))/B \right\rfloor = \sum_{1 \leq i_1 \leq k} \lfloor \chi((w_{i_1}, w_{i_2}))/B \rfloor$, where the third equality holds because we have $\chi((w_{i_1}, w_{i_2}))/B - \lfloor \chi((w_{i_1}, w_{i_2}))/B \rfloor \leq M/B < 1/k$. The definition of $X_{(w_{i_1},w_{i_2})}$ implies that $\lfloor \chi((w_{i_1}, w_{i_2}))/B \rfloor$ is an integer from X. Therefore, the equation above states the the sum of k integers from X is exactly $kx_{j_{i_2}}$. Since X is a k-non-averaging set, this is only possible if these k integers are all equal to $x_{j_{i_2}}$. Therefore, we have shown that $\chi((w_{i_1}, w_{i_2}))$ modulo B is exactly $x_{i_{j_1}}$ and $\lfloor \chi((w_{i_1}, w_{i_2}))/B \rfloor$ is exactly $x_{j_{i_2}}$, proving that $\chi((w_{i_1}, w_{i_2})) = x_{j_{i_1}} + Bx_{j_{i_2}}$ indeed holds. ◀

15.2 Constrained Closed Walk and ATSP

To make the hardness proof for ATSP cleaner, we first prove hardness for the variant of the problem, where instead of optimizing the length of the tour, the only constraint is that certain vertices cannot be visited more than once.

Constrained Closed Walk: Given an unweighted directed graph G and set $U \subseteq V(G)$ of vertices, find a closed walk (of any length) that visits each vertex at least once and visits each vertex in U exactly once.

There is a simple reduction from Constrained Closed Walk to ATSP that preserves treewidth.

▶ **Lemma 87.** *An instance of* Constrained Closed Walk *on an unweighted directed graph D can be reduced in polynomial time to an instance of* ATSP *with polynomially bounded positive integer weights on an edge-weighted version D^* of D.*

Proof. It is easy to see that if we assign weight 1 to every edge (u, v) with $v \in U$ and weight 0 to every other edge, then the Constrained Closed Walk instance has a solution if and only if the resulting weighted graph has closed walk of length at most $|U|$ (or, equiavelently, less than $|U| + 1$) visiting every vertex. To ensure that every weight is positive, let us replace every weight 0 with weight $\epsilon := 1/(2n^2)$. As a minimum solution of ATSP contains at most n^2 edges, this modification increases the minimum cost by at most $1/2$. Thus it remains true that the Constrained Closed Walk instance has a solution if and only if there is a closed walk of length less than $|U| + 1$ visiting every vertex. Finally, to ensure that every cost is integer, we multiply each of them by $2n^2$. ◀

The rest of the section is devoted to giving a lower bound for Constrained Closed Walk. The lower bound proof uses certain gadgets in the construction of the instances. Formally, we define a *gadget* to be a graph with a set of distinguished vertices called *external vertices;* every other vertex is *internal.* To avoid degenerate situations, we always require that the external vertices of a gadget are independent and each external vertex has either indegree 0 or outdegree 0 in the gadget; in particular, this implies that a path between two external vertices contains no other external vertex. Also, this implies that there is no closed walk containing an external vertex.

We say that a set $\mathcal{P}$ of paths of the gadget *satisfies* a gadget if (1) both endpoints of each path are external vertices and (2) every internal vertex of the gadget is visited by exactly one path in $\mathcal{P}$. If a path $P \in \mathcal{P}$ connects two external vertices of a gadget, then we define the *type* of P to be the (ordered) pair of its endpoints. If $\mathcal{P}$ satisfies the gadget, then we define the *type* of $\mathcal{P}$ to be the multiset of the types of the paths in $\mathcal{P}$. For brevity, we use notation such as $a \times (v_1, v_2) + b \times (v_3, v_4)$ to denote the type that contains a times the pair (v_1, v_2) and b times the pair (v_3, v_4). For a gadget H, we let the set $\mathcal{T}(H)$ contain every possible type of a set $\mathcal{P}$ of paths satisfying H.

We construct gadgets where we can exactly tell the type of the collections of paths that can satisfy the gadget, that is, the set $\mathcal{T}(H)$ is of a certain form. In the first gadget, we have a simple choice between one path or a specified number of paths.

▶ **Lemma 88.** *For every $s \geq 1$, we can construct in time polynomial in n a gadget H_s with the following properties:*

1. *H_s has four external vertices a_{in}, a_{out}, b_{in}, and b_{out}.*
2. *H_s minus its external vertices has constant pathwidth.*
3. *$\mathcal{T}(H_s)$ contains exactly two types: the type $(b_{\text{in}}, b_{\text{out}})$ and the type $s \times (a_{\text{in}}, a_{\text{out}})$ (in other words, the gadget can be satisfied by a path from b_{in} to b_{out}), can be satisfied by a collection of s paths from a_{in} to a_{out}), but cannot be satisfied by any other type of collection of paths).*

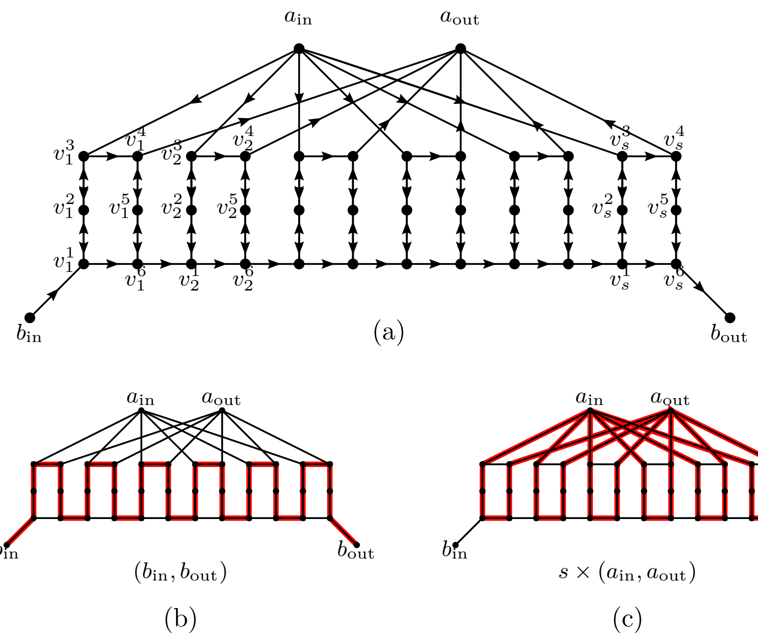


Figure 6 The gadget of Lemma 88 with two collections of paths satisfying it.

Proof. The gadget H_s has $6s$ internal vertices v_j^i ($1 \le i \le 6$, $1 \le j \le s$) connected as shown in Figure 6(a). Additionally, we introduce the edges (b_{in}, v_1^1), $(v_s^6, b_{\text{out}}$, and for every $1 \le j \le s$, the edges (a_{in}, v_j^3) and (v_j^4, a_{out}). It is clear that statement (2) holds: H_s minus its vertices is a graph with constant pathwidth.

This gadget can be satisfied by a path from b_{in} to b_{out} (see Figure 6(b)) and also by a collection of s paths where the j-th path is a_{in}, v_j^3, v_j^2, v_j^1, v_j^6, v_j^5, v_j^4, a_{out} (see Figure 6(c)). To complete the proof of statement (3), we need to show that if $\mathcal{P}$ satisfies H_s, then $\mathcal{P}$ is one of these two types. The basic observation is that if a path in $\mathcal{P}$ contains v_j^2, then it has to contain v_j^1 and v_j^3 as well (as each internal vertex is visited exactly once), hence the three vertices v_j^1, v_j^2, v_j^3 have to appear on the same path of $\mathcal{P}$. The same is true for the vertices v_j^4, v_j^5, v_j^6. Suppose that $\mathcal{P}$ contains a path P starting at b_{in}. Then its next vertex is v_1^1, which should be followed by v_1^2 and v_1^3 by the argument above. The next vertex is v_1^4 (the only outneighbor of v_j^3 not yet visited), which is followed by v_1^5 and v_1^6. Now the next vertex is v_2^1, the only outneighbor of v_1^6 not yet visited. With similar arguments, we can show that P is exactly of the form shown in Figure 6(b), hence $\mathcal{P}$ contains only this path, and $\mathcal{P}$ is of type $\{(b_{\text{in}}, b_{\text{out}})\}$.

Suppose now that $\mathcal{P}$ does not contain a path starting at b_{in}. Then the only way to reach vertex v_1^1 is with a path starting as a_{in}, v_1^3, v_1^2, v_1^1. This has to be followed by the unique outneighbor v_1^6 of v_1^1 that was not yet visited. This means that the path contains also v_1^5 and v_1^4, which has to be followed by a_{out}. Then with similar arguments, we can show for every $j \ge 2$ that v_j^1 is visited by the path a_{in}, v_j^3, v_j^2, v_j^1, v_j^6, v_j^5, v_j^4, a_{out}. This means that $|\mathcal{P}| = s$ and the type of $\mathcal{P}$ is $s \times (a_{\text{in}}, a_{\text{out}})$. ◀

▶ **Lemma 89.** *Let X be a set of n positive integers, each at most M and let $S = \sum_{x \in X} x$. In time polynomial in n and M, we can construct a gadget H_X with the following properties:*

1. *H_X has four external vertices a_{in}, a_{out}, c_{in}, and c_{out}.*
2. *H_X minus its external vertices has constant pathwidth.*

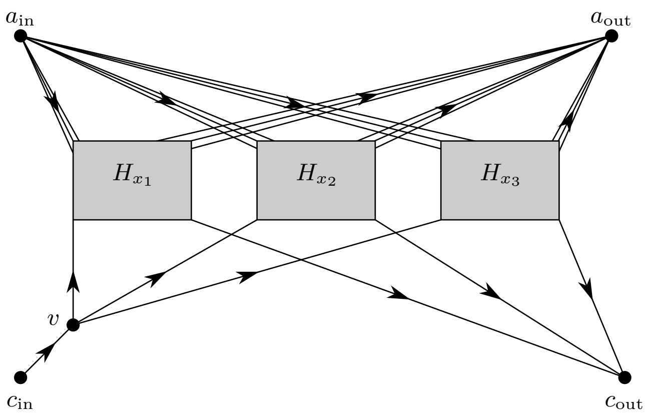


Figure 7 The gadget H_X of Lemma 89 for a set $X = \{x_1, x_2, x_3\}$ of three integers. The gray rectangles represent the *internal* vertices of the three gadgets H_{x_1}, H_{x_2}, and H_{x_3}.

3. *$\mathcal{T}(H_s)$ contains exactly $|X|$ types: for every $x \in X$, it contains the type $(c_{\text{in}}, c_{\text{out}}) + (S - x) \times (a_{\text{in}}, a_{\text{out}})$.*

Proof. The gadget H_X is constructed the following way (see Figure 7). Let us introduce an internal vertex v and the edge (c_{in}, v). For every $x \in X$, let us introduce a copy of H_x defined by Lemma 88 where a_{in}, a_{out}, v, c_{out} of H_X play the role of a_{in}, a_{out}, b_{in}, b_{out}, respectively. If we remove the four external vertices of H_X, then we get a graph with constant pathwidth: if we remove one more vertex, v, then we get the disjoint union of internal vertices of the gadgets H_x's, which have constant pathwidth by Lemma 88.

For every $x \in X$, the gadget can be satisfied by the following collection of paths. The copy of H_x in H_X can be satisfied by a path from v to c_{out}, which can be extended with the edge (c_{in}, v) to a path from c_{in} to c_{out}. For every $x' \in X$, $x' \neq x$, we can satisfy the copy of $H_{x'}$ in H_X by a collection of x' paths from a_{in} to a_{out}. This way, we constructed a collection $\mathcal{P}$ of paths satisfying H_X that consists of a single path of type $(c_{\text{in}}, c_{\text{out}})$ and exactly $\sum_{x' \in X \setminus \{x\}} x' = S - x$ paths of type $(a_{\text{in}}, a_{\text{out}})$.

To complete the proof of statement (3), consider a collection $\mathcal{P}$ of paths satisfying H_X. Let P be the unique path of $\mathcal{P}$ visiting vertex v. The vertex of P after v is has to be an internal vertex of the copy of H_x for some $x \in X$ (here we use that the external vertices of the gadget H_x are independent, hence v cannot be followed by any of a_{in}, a_{out}, and c_{out}). As v was identified with vertex b_{in} of H_x, Lemma 88 implies that P visits every internal vertex of this copy of H_x and leaves H_x at its vertex b_{out}, which was identified with c_{out}. Consider now some $x' \in X$ with $x' \neq x$. Vertex b_{in} of $H_{x'}$ was identified with v, path P is the only path of $\mathcal{P}$ visiting v, and P does not visit any internal vertex of $H_{x'}$. Therefore, by Lemma 88, the internal vertices of $H_{x'}$ are visited by exactly x' paths of type $(a_{\text{in}}, a_{\text{out}})$. Thus $\mathcal{P}$ contains one path of type $(c_{\text{in}}, c_{\text{out}})$ and exactly $\sum_{x' \in X \setminus \{x\}} x' = S - x$ paths of type $(a_{\text{in}}, a_{\text{out}})$. ◀

▶ **Lemma 90.** *Assuming ETH, there is no $f(p)n^{o(p)}$ time algorithm for* Constrained Closed Walk *on graphs of pathwidth at most p for any computable function f.*

Proof. The proof is by reduction from Edge Balancing on a directed graph D with k vertices $w_1, \ldots, w_k$. We construct a Constrained Closed Walk instance on a directed graph D^* the following way. First, let us introduce the vertices $w_1, \ldots, w_k$ into D^*, as well as two auxiliary vertices c_{in} and c_{out}. For every edge $e = (w_{i_1}, w_{i_2}) \in E(D)$ with a set X_e

of integers associated to it in the Edge Balancing instance, we construct a copy of the gadget H_{X_e} defined by Lemma 89 and identify external vertices a_{in}, a_{out}, c_{in}, c_{out} of the gadget H_{X_e} with vertices w_{i_1}, w_{i_2}, c_{in}, c_{out} of D^*, respectively. Let $S_e = \sum_{x \in X_e} x$ for every edge $e \in V(D)$, let $S_i^+ = \sum_{e \in \delta_D^+(w_i)} S_e$, let $S_i^- = \sum_{e \in \delta_D^-(w_i)} S_e$, and let $S^* = \sum_{e \in E(D)} X_e = \sum_{i=1}^k S_i^+ = \sum_{i=1}^k S_i^-$. We further extend D^* the following way.

1. For every $1 \le i \le k$, we introduce a set $\mathcal{P}_i^+$ of S_i^+ paths of length two from c_{in} to w_i (that is, each of these paths consists of vertex c_{in}, vertex w_i, and one extra newly introduced vertex).
2. For every $1 \le i \le k$, we introduce a set $\mathcal{P}_i^-$ of S_i^- paths of length two from w_i to c_{out}.
3. We introduce a set $\mathcal{P}^*$ of $S^* + |E(D)|$ paths of length two from c_{out} to c_{in}.

Let $Z := \{w_1, \ldots, w_k, c_{\text{in}}, c_{\text{out}}\}$; note that Z form an independent set in G^* (as the external vertices of each gadget are independent). We define $U := V(D^*) \setminus Z$ to be the set of vertices that have to be visited exactly once. This completes the description of the reduction.

Observe that if we remove Z from D^*, then what remains is the disjoint union of the internal vertices of the gadgets H_{X_e}, which have constant pathwidth by Lemma 89. As removing a vertex can decrease pathwidth at most by one, it follows that D^* has pathwidth $|Z| + O(1) = O(k)$. Thus if we are able to show that the constructed instance D^* of Constrained Closed Walk is a yes-instance if and only if D is a yes-instance of Edge Balancing, then this implies that an $f(p)n^{o(p)}$ time algorithm for Constrained Closed Walk on graphs of pathwidth p can be used to solve Edge Balancingon k vertex graphs in time $(k)n^{o(k)}$, which would contradict ETH by Lemma 86.

Balanced assignment $\chi \Rightarrow$ closed walk. Suppose that balanced assignment $\chi : E(D) \to \mathbb{Z}^+$ is a solution to the Edge Balancing instance. For every $e = (w_{i_1}, w_{i_2}) \in E(D)$, the construction of the gadget H_{X_e} implies that H_{X_e} can be satisfied by a collection $\mathcal{P}_e$ of paths having type $(c_{\text{in}}, c_{\text{out}}) + (S_e - \chi(e)) \times (w_{i_1}, w_{i_2})$. Let $\mathcal{P}$ be a collection of paths that is the union of the set $\mathcal{P}^*$, the sets $\mathcal{P}_i^+$ and $\mathcal{P}_i^-$ for $1 \le i \le k$, and the set $\mathcal{P}_e$ for $e \in E(G)$. Observe that every vertex of U is contained in exactly one path in $\mathcal{P}$ and the paths in $\mathcal{P}$ are edge disjoint. Let H^* be the subgraph of D^* formed by the union of every path in $\mathcal{P}$. It is easy to see that H^* is connected: every path in $\mathcal{P}$ has endpoints in Z and the paths in $\mathcal{P}^*$, $\mathcal{P}_i^-$, $\mathcal{P}_i^+$ ensure that every vertex of Z is in the same component of H^*. It is also clear that every vertex of U has indegree and outdegree exactly 1, as each vertex in U is visited by exactly one path in $\mathcal{P}$. We show below that every vertex of Z is balanced in H^* (its indegree equals its outdegree). If this is true, then H^* has a closed Eulerian walk, which gives a closed walk in G^* visiting every vertex at least once and every vertex in U exactly once, what we had to show.

The endpoints of every path in $\mathcal{P}$ are in Z, hence every vertex of U is balanced in H^* (in particular has indegree and outdegree exactly 1). Consider now a vertex w_i.

- For every $e \in \delta_D^+(w_i)$, the set $\mathcal{P}_e$ contains $S_e - \chi(e)$ paths starting at w_i.
- For every $e \in \delta_D^-(w_i)$, the set $\mathcal{P}_e$ contains $S_e - \chi(e)$ paths ending at w_i.
- The set $\mathcal{P}_i^+$ contains S_i^+ paths ending at w_i.
- The set $\mathcal{P}_i^-$ contains S_i^- paths starting at w_i.

As these paths are edge disjoint, the difference between the outdegree and the indegree of w_i in H^* is $(S_i^- + \sum_{e \in \delta_D^+(w_i)}(S_e - \chi(e))) - (S_i^+ + \sum_{e \in \delta_D^-(w_i)}(S_e - \chi(e))) = (S_i^- + S_i^+ - \chi(\delta_D^+(w_i))) - (S_i^+ + S_i^- - \chi(\delta_D^-(w_i))) = 0$, since χ is balanced at w_i. Consider now vertex c_{in}.

- For every $1 \le i \le k$, the set $\mathcal{P}_i^+$ contains S_i^+ paths starting at c_{in}.
- For every $e \in E(D)$, the set $\mathcal{P}_e$ contains one path starting at c_{in}.
- The set $\mathcal{P}^*$ contains $S^* + |E(D)|$ paths ending at c_{in}.

It follows that c_{in} is balanced in H^* with indegree and outdegree exactly $S^* + |E(D)| = \sum_{i=1}^{k} S_i^+ + |E(D)|$ and a similar argument shows the same for c_{out}. Thus we have shown that the Constrained Closed Walk instance has a solution.

Closed walk $\Rightarrow$ balanced assignment χ. For the reverse direction, suppose that the constructed Constrained Closed Walk instance has a solution (a closed walk W). The closed walk can be split into a collection $\mathcal{P}$ of walks with endpoints in Z and every internal vertex in U. In fact, these walks are paths: (1) as each vertex of U is visited only once, the internal vertices of each walk are distinct, (2) the walk cannot be a cycle, since we have stated earlier that no gadget has a cycle through an external vertex. When defining the sets $\mathcal{P}^*$, $\mathcal{P}_i^+$, $\mathcal{P}_i^-$, we introduced a large number of vertices into D^* with indegree and outdegree 1. The fact that these vertices are visited implies that $\mathcal{P}$ has to contain the set $\mathcal{P}^*$ and the sets $\mathcal{P}_i^+$ and $\mathcal{P}_i^-$ for every $1 \le i \le k$. Moreover, every path of $\mathcal{P}$ not in these sets contains an internal vertex of some gadget H_{X_e} (here we use that Z is independent) and a path of $\mathcal{P}$ cannot contain the internal vertices of two gadgets (as this would imply that it has an internal vertex in Z). Therefore, the remaining paths can be partitioned into sets $\mathcal{P}_e$ for $e \in E(D)$ such that the internal vertices of H_{X_e} are used only by the paths in $\mathcal{P}_e$. This means that the set $\mathcal{P}_e$ satisfies gadget H_{X_e}. If $e = (w_{i_1}, w_{i_2})$, then it follows by Lemma 89 that $\mathcal{P}_e$ has type $(c_{\text{in}}, c_{\text{out}}) + (S_e - \chi(e))(w_{i_1}, w_{i_2})$ for some integer $\chi(e) \in X_e$. In particular, this means that $\mathcal{P}_e$ contains $S_e - \chi(e)$ paths starting at w_{i_1} and the same number of paths ending at w_{i_2}.

We claim that χ form a solution of the Edge Balancing problem. Consider a vertex w_i. Taking into account the contribution of the paths in $\mathcal{P}_i^-$ and $\mathcal{P}_e$ for $e \in \delta_D^+(w_i)$, we have that the outdegree of w_i in the walk W is exactly $S_i^- + \sum_{e \in \delta_D^+(w_i)} (S_e - \chi(e)) = S_i^+ + S_i^- - \chi(\delta_D^+(w_i))$. Taking into account the contribution of the paths in $\mathcal{P}_i^+$ and $\mathcal{P}_e$ for $e \in \delta_D^-(w_i)$, we have that the indegree of w_i in the walk W is exactly $S_i^+ + \sum_{e \in \delta_D^-(w_i)} (S_e - \chi(e)) = S_i^- + S_i^+ - \chi(\delta_D^-(w_i))$. As the indegree of w_i in W is clearly the same as its outdegree, these two values have to be equal. This is only possible if $\chi(\delta_D^+(w_i)) = \chi(\delta_D^-(w_i))$, that is χ is balanced at w_i. As this is true for every $1 \le i \le k$, it follows that the Edge Balancing instance has a solution. ◀

References

1 Nima Anari and Shayan Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric tsp. In *55th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2015.

2 Arash Asadpour, Michel X Goemans, Aleksander Madry, Shayan Oveis Gharan, and Amin Saberi. An $O(\log n/\log\log n)$-approximation Algorithm for the Asymmetric Traveling Salesman Problem. In *SODA*, volume 10, pages 379–389. SIAM, 2010.

3 Markus Bläser. A new approximation algorithm for the asymmetric tsp with triangle inequality. *ACM Transactions on Algorithms (TALG)*, 4(4):47, 2008.

4 Moses Charikar, Michel X Goemans, and Howard Karloff. On the integrality ratio for asymmetric tsp. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 101–107. IEEE, 2004.

5 Jianer Chen, Xiuzhen Huang, Iyad A Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006.

6 Holger Dell and Dániel Marx. Kernelization of packing problems. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 68–81. SIAM, 2012.

7 Reinhard Diestel. *Graph theory {graduate texts in mathematics; 173}*. Springer-Verlag Berlin and Heidelberg GmbH & amp, 2000.

8 Jeff Erickson and Anastasios Sidiropoulos. A near-optimal approximation algorithm for asymmetric tsp on embedded graphs. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 130. ACM, 2014.

9 Uriel Feige and Mohit Singh. Improved approximation ratios for traveling salesperson tours and paths in directed graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 104–118. Springer, 2007.

10 Fedor V Fomin, Petr A Golovach, Daniel Lokshtanov, and Saket Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM Journal on Computing*, 43(5):1541–1563, 2014.

11 Alan M Frieze and Giulia Galbiati. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982.

12 Alan M. Frieze, Giulia Galbiati, and Francesco Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982. `doi:10.1002/net.3230120103`.

13 Shayan Oveis Gharan and Amin Saberi. The asymmetric traveling salesman problem on graphs with bounded genus. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 967–975. SIAM, 2011.

14 F. Harary. *Graph Theory*. Addison-Wesley Series in Mathematics. Perseus Books, 1994.

15 Michael Held and Richard Karp. The traveling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

16 Michael Held and Richard M Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970.

17 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013.

18 Haim Kaplan, Moshe Lewenstein, Nira Shafrir, and Maxim Sviridenko. Approximation algorithms for asymmetric tsp by decomposing directed regular multigraphs. *Journal of the ACM (JACM)*, 52(4):602–626, 2005.

19 Ken-ichi Kawarabayashi and Bojan Mohar. Some recent progress and applications in graph minor theory. *Graphs and Combinatorics*, 23(1):1–46, 2007.

20 László Lovász. Graph minor theory. *Bulletin of the American Mathematical Society*, 43(1):75–86, 2006.

21 A. Malnič and B. Mohar. Generating locally cyclic triangulations of surfaces. *Journal of Combinatorial Theory, Series B*, 56(2):147–164, 1992.

22 Carsten Thomassen. Embeddings of graphs with no short noncontractible cycles. *Journal of Combinatorial Theory, Series B*, 48(2):155–177, 1990.

Planar Matching in Streams Revisited*

Andrew McGregor[1] and Sofya Vorotnikova[2]

1 **College of Information and Computer Sciences, University of Massachusetts, Amherst, USA**
mcgregor@cs.umass.edu

2 **College of Information and Computer Sciences, University of Massachusetts, Amherst, USA**
svorotni@cs.umass.edu

Abstract

We present data stream algorithms for estimating the size or weight of the maximum matching in low arboricity graphs. A large body of work has focused on improving the constant approximation factor for general graphs when the data stream algorithm is permitted $O(n \operatorname{polylog} n)$ space where n is the number of nodes. This space is necessary if the algorithm must return the matching. Recently, Esfandiari et al. (SODA 2015) showed that it was possible to estimate the maximum cardinality of a matching in a planar graph up to a factor of $24 + \epsilon$ using $O(\epsilon^{-2} n^{2/3} \operatorname{polylog} n)$ space. We first present an algorithm (with a simple analysis) that improves this to a factor $5 + \epsilon$ using the same space. We also improve upon the previous results for other graphs with bounded arboricity. We then present a factor 12.5 approximation for matching in planar graphs that can be implemented using $O(\log n)$ space in the adjacency list data stream model where the stream is a concatenation of the adjacency lists of the graph. The main idea behind our results is finding "local" fractional matchings, i.e., fractional matchings where the value of any edge e is solely determined by the edges sharing an endpoint with e. Our work also improves upon the results for the *dynamic* data stream model where the stream consists of a sequence of edges being inserted and deleted from the graph. We also extend our results to weighted graphs, improving over the bounds given by Bury and Schwiegelshohn (ESA 2015), via a reduction to the unweighted problem that increases the approximation by at most a factor of two.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases data streams, planar graphs, arboricity, matchings

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.17

1 Introduction

A large body of work has focused on finding better approximation algorithms for finding large graph matchings in the data stream model [1, 11, 24, 26, 7, 8, 13, 19, 18, 15, 21, 22, 4, 20, 17]. In this model, the edges of an input graph on n nodes arrive in an arbitrary order and the algorithm has a limited amount of memory available. For a survey of graph algorithms in this model, see [25]. Specifically, a sequence of papers have presented algorithms using $O(n \operatorname{polylog} n)$ bits of space that have steadily reduced the best known approximation ratio for maximum weighted matching: Feigenbaum et al. [13] initially presented a 6 approximation; McGregor [24] then presented a 5.828 approximation; this was reduced to 5.585 by Zelke [26]; and then to 4.911 by Epstein et al. [11]; and the best known result is a 4 approximation

* Supported by NSF CAREER Award CCF-0953754 and CCF-1320719 and a Google Faculty Research Award.

due to Crouch and Stubbs [7]. The best known result for maximum cardinality matching is a trivial 2 factor that follows by constructing a greedy matching. Konrad et al. [21] showed that this can be slightly improved if the edges are ordered randomly. Kapralov [18] proved a lower bound of $e/(e-1) \approx 1.58$ on the best possible approximation factor when using only $O(n \operatorname{polylog} n)$ space. Note that all the above algorithms return the large matching rather than just estimating its weight.

A natural question is whether constant approximation is possible using $o(n)$ space if we only need to estimate the weight of the matching. Recently, Esfandiari et al. [12] showed the surprising result that it was indeed possible in the case of planar graphs and more generally, *bounded arboricity graphs.* Recall that a graph G has arboricity α if the set of edges of G can be partitioned into at most α forests. For example, a planar graph has arboricity $\alpha = 3$. Esfandiari et al. presented a $(5\alpha+9)(1+\epsilon)$ approximation using $O(\alpha\epsilon^{-2}n^{2/3} \operatorname{polylog} n)$ space. In the case of planar graphs this corresponds to a $24+\epsilon$ approximation. Very recently Chitnis et al. [5] and Bury and Schwiegelshohn [4], showed that the same approximation was possible in the dynamic graph model where the stream consists of edges being added and deleted from the underlying graph. Bury and Schwiegelshohn [4] also showed that it was possible to extend the result to weighted graphs but with an $O(\alpha^4)$ approximation factor. All of these results rely on an interesting structural result proved by Esfandiari et al. [12] that relates the size of the maximum cardinality matching in a graph G of arboricity α to the number of nodes of "high" degree and the number of edges whose endpoints are both "low" degree. Specifically:

▶ **Theorem 1** (Esfandiari et al. [12]). *Let* $\operatorname{match}(G)$ *be the size of the maximum cardinality matching in* G*. Then*

$$\operatorname{match}(G)/2 \leq (h+s)/2 \leq \max(h,s) \leq (2.5\alpha + 4.5)\operatorname{match}(G)$$

where h *is the number of nodes with degree greater than* $2\alpha+3$ *and* s *is the number of edges that remain after all such nodes are removed.*

The result then reduces the problem of estimating the matching to estimating the quantities h and s. A similar, but weaker bound, is implicit in Czygrinow et al. [9].

1.1 Our Results

The first contribution of this paper is to identify a new quantity that a) yields tighter bounds for $\operatorname{match}(G)$ and b) can be approximated in the data stream model.

▶ **Theorem 2** (Structural Result). *For an edge* $e = \{u,v\}$ *define* $x_e = \min\left(\frac{1}{\deg(u)}, \frac{1}{\deg(v)}, \frac{1}{\alpha+1}\right)$. *Then,*

$$\operatorname{match}(G) \leq (\alpha+1)\sum_{e\in E} x_e \leq (\alpha+2)\operatorname{match}(G)\,.$$

For example, for a planar graph $\alpha = 3$ and hence $\sum_{e\in E} x_e$ determines $\operatorname{match}(G)$ up to a factor of 5. For a bipartite planar graph (for such graphs, $\alpha = 2$) the result can be further improved to a factor of 3. The proof of Theorem 2 can be found in Section 2.2 and has the advantage of being conceptually simpler than the proof of Theorem 1. The main idea is to prove the result via consideration of fractional matchings, specifically "local" fractional matching where the value of any edge e can be determined by only considering the edges incident to e. We also show a result for local fractional matchings for weighted graphs.

Using Theorem 2, we show that $\text{match}(G)$ on unweighted graphs can be approximated up to a factor $(\alpha+2)(1+\epsilon)$ using $O(\epsilon^{-2}\alpha n^{2/3}\,\text{polylog}\,n)$ bits of space. Furthermore, this result can be generalized to weighted graphs with approximation factor $2(\alpha+2)(1+\epsilon)$ and to the dynamic graph stream model with a slight increase in space. We also show that it is possible to estimate $\text{match}(G)$ up to a factor $(\alpha+2)^2/2$ using only the degree sequence of G. This result immediately leads to a $O(\log n)$ space algorithm in the adjacency list stream model where the stream is a concatenation of the adjacency lists of the graph. The result can also be generalized to weighted graphs while losing only an additional factor of 2.

2 Graph Properties

In this section we present a variety of results relating the size or weight of a maximum matching in a low arboricity graph to "simpler" quantities. We start with some necessary preliminaries about fractional matchings.

2.1 Preliminaries

Define the fractional matching polytope for a graph G as:

$$\mathsf{FM}(G) = \{\mathbf{x} \in \mathbb{R}^E : x_e \geq 0 \text{ for all } e \in E, \sum_{e\in E: u\in e} x_e \leq 1 \text{ for all } u \in V\}\,.$$

We say any $\mathbf{x} \in \mathsf{FM}(G)$ is a fractional matching. The size of this fractional matching is $\sum_{e\in E} x_e$ and for a graph where edge e has weight w_e, the weight of the matching is $\sum_{e\in E} w_e x_e$. A standard result on fractional matching is that the maximum size of a fractional matching is at most a factor 3/2 larger than the maximum size of an (integral) matching. We will also make use of the following lemma which is a simple corollary of Edmonds Matching Polytope theorem [10].

▶ **Lemma 3.** *For $U \subset V$, let $G[U]$ denote the induced subgraph on U. Let $\mathbf{x} \in \mathsf{FM}(G)$ and suppose there exist $\lambda_3, \lambda_5, \lambda_7 \ldots$ such that*

$$\forall U \subset V \textit{ where } |U| \in \{3,5,7,\ldots\}\ , \quad \sum_{e\in G[U]} x_e \leq \lambda_{|U|}\left(\frac{|U|-1}{2}\right)\,.$$

Then for any edge weights $\{w_e\}_{e\in E}$,

$$\sum_{e\in E} w_e x_e \leq \max(1,\lambda_3,\lambda_5,\ldots)\,\text{match}(G)$$

where $\text{match}(G)$ is the weight of the maximum weighted (integral) matching.

Proof. By Edmonds theorem, $\text{match}(G) = \max_{\mathbf{z}\in\mathsf{IM}(G)} \sum_e w_e z_e$ where

$$\mathsf{IM}(G) = \{\mathbf{x} \in \mathbb{R}^E : x_e \geq 0 \text{ for all } e \in E, \sum_{e\in E: u\in e} x_e \leq 1 \text{ for all } u \in V,$$
$$\sum_{e\in G[U]} x_e \leq \left(\frac{|U|-1}{2}\right) \text{ for all } U \subset V \text{ of odd size}\}\,.$$

But $\frac{\mathbf{x}}{\max(1,\lambda_3,\lambda_5,\ldots)} \in \mathsf{IM}(G)$ and so $\sum_{e\in E} w_e x_e \leq \max(1,\lambda_3,\lambda_5,\ldots)\,\text{match}(G)$ as required. ◀

For the streaming applications we will be interested in fractional matchings that can be computed locally.

▶ **Definition 4.** For a given graph G, we say a fractional matching $\mathbf{x} \in \mathsf{FM}(G)$ is *local* if every x_e is only a function of the edges (and their weights in the case of a weighted graph) that share an end point with e.

2.2 Local Fractional Matching

Define $\mathbf{x} \in \mathbb{R}^E$ where for $e = \{u, v\} \in E$, we set

$$x_e = \min\left(\frac{1}{\deg(u)}, \frac{1}{\deg(v)}, \frac{1}{\alpha+1}\right).$$

The next two theorems show that $\mathbf{x}$ is a local fractional matching and

$$\frac{1}{\alpha+1} \cdot \operatorname{match}(G) \le \operatorname{score}(\mathbf{x}) \le \frac{\alpha+2}{\alpha+1} \cdot \operatorname{match}(G)$$

where $\operatorname{score}(\mathbf{x}) = \sum_e x_e$. This proves Theorem 2 and we note that the upper bound can be improved slightly if α is even. In Section 3.1, we show that it is possible to efficiently estimate $\operatorname{score}(\mathbf{x})$ in the data stream model.

▶ **Theorem 5.** $\mathbf{x} \in \mathsf{FM}$ *and*

$$\frac{\operatorname{score}(\mathbf{x})}{\operatorname{match}(G)} \le \begin{cases} \frac{\alpha+2}{\alpha+1} & \text{if } \alpha \text{ odd,} \\ \frac{\alpha+3}{\alpha+2} & \text{if } \alpha \text{ even.} \end{cases}$$

Furthermore, if G is bipartite then $\operatorname{score}(\mathbf{x}) \le \operatorname{match}(G)$.

Proof. First note that $x_e \ge 0$ for each $e \in E$ and for any $u \in V$,

$$\sum_{e \in E: u \in e} x_e \le \sum_{e \in E: u \in e} 1/\deg(u) = 1.$$

and hence $\mathbf{x} \in \mathsf{FM}$. The bound for bipartite graphs follows because the maximum size of a fractional matching in a bipartite graph equals the maximum size of an integral matching. For the rest of the result, we appeal to Lemma 3. Since $\mathbf{x} \in \mathsf{FM}$, it is simple to show that $\mathbf{x}$ satisfies the conditions of the lemma with $\lambda_t \le t/(t-1)$; this follows because $\sum_{e \in G[U]} x_e \le |U|/2$ for any $\mathbf{x} \in \mathsf{FM}$. Furthermore, since there are at most $\binom{|U|}{2}$ edges in $G[U]$ and $x_e \le 1/(\alpha+1)$ for all e,

$$\sum_{e \in G[U]} x_e \le \binom{|U|}{2} \frac{1}{\alpha+1} = \frac{|U|-1}{2} \cdot \frac{|U|}{\alpha+1}.$$

Therefore, $\lambda_t \le \min\left(t/(t-1), t/(\alpha+1)\right)$. Consequently,

$$\max_{t \text{ odd}} \lambda_t = \begin{cases} \frac{\alpha+2}{\alpha+1} & \text{if } \alpha \text{ odd,} \\ \frac{\alpha+3}{\alpha+2} & \text{if } \alpha \text{ even.} \end{cases}$$ ◀

We next bound $\operatorname{score}(\mathbf{x})$ in terms of the number of high degree vertices and edges that are not incident to high degree vertices. As observed in previous work, these two quantities can then easily be related the size of the maximum matching.

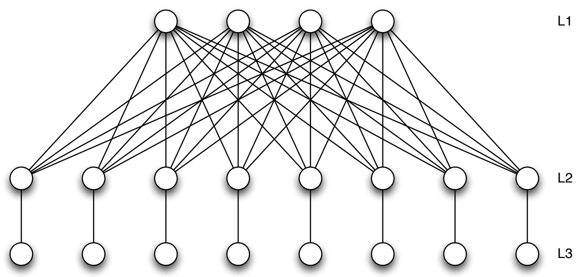


Figure 1 A tight example for Theorem 6. Let L_1 consist of α nodes whereas L_2 and L_2 consist of $n \gg \alpha$ nodes. The edges are a complete bipartite graph of L_1 and L_2 and a matching between L_2 and L_3. Then $\mathrm{score}(\mathbf{x}) = \alpha n \times 1/n + n \times 1/(\alpha+1)$ and $\mathrm{match}(G) = n$. Hence $\mathrm{match}(G)/\mathrm{score}(\mathbf{x})$ tends to $\alpha + 1$ as n tends to infinity.

▶ **Theorem 6.** *Let h be the number of "heavy" nodes with degree at least $\alpha + 2$ and s be the number of "shallow" edges whose endpoints are both not heavy. Then,*

$$\mathrm{score}(\mathbf{x}) \geq 2h/(\alpha+2) + s/(\alpha+1)\,.$$

Furthermore, $\mathrm{match}(G) \leq (\alpha+1)\,\mathrm{score}(\mathbf{x})$.

Proof. Let d_i be the degree of node i and assume $d_1 \geq d_2 \geq d_3 \geq \ldots$. Let $b_i = |\{j < i : \{i,j\} \in E\}|$ and $c_i = |\{i < j : \{i,j\} \in E\}|$, i.e., the number of neighbors of node i that have higher or lower degree respectively than node i where ties are broken by the ordering supposed in the above line. Consider labeling an edge e with weight x_e where we first label edges incident to node 1, then the (remaining unlabeled) edges incident to node 2, etc. Then $c_1 = d_1$ edges get labeled with $\min(1/d_1, 1/(\alpha+1))$, c_2 edges get labeled with $\min(1/d_2, 1/(\alpha+1))$, c_3 edges get labeled with $\min(1/d_3, 1/(\alpha+1))$ etc. Let $\theta = \alpha + 2$, then

$$\begin{aligned}
\mathrm{score}(\mathbf{x}) &= \sum_i c_i \min(1/d_i, 1/(\alpha+1)) \\
&= \sum_{i:d_i\geq\theta} c_i/d_i + \sum_{i:d_i\leq\theta-1} c_i/(\alpha+1) \\
&= h - \sum_{i:d_i\geq\theta} b_i/d_i + \sum_{i:d_i\leq\theta-1} c_i/(\alpha+1) \\
&\geq h - (\sum_{i:d_i\geq\theta} b_i)/\theta + (\sum_{i:d_i\leq\theta-1} c_i)/(\alpha+1)
\end{aligned}$$

Note that $\sum_{i:d_i\geq\theta} b_i$ is the number of edges in the induced subgraph on heavy nodes. This is at most αh because these edges in this induced subgraph can be partitioned into at most α forests. Similarly, $\sum_{i:d_i\leq\theta-1} c_i$ is the number of shallow edges. Therefore

$$\mathrm{score}(\mathbf{x}) \geq h(1-\alpha/\theta) + s/(\alpha+1) = 2h/(\alpha+2) + s/(\alpha+1)$$

as required. Note that $h + s \geq \mathrm{match}(G)$ because every edge in a matching is either shallow or has at least one heavy node as an endpoint. Therefore

$$\mathrm{score}(\mathbf{x}) \geq (h+s)/(\alpha+1) \geq \mathrm{match}(G)/(\alpha+1)\,. \qquad ◀$$

See Figure 1 for an example that shows that the above theorem is tight.

2.3 Local Fractional Matchings for Weighted Graphs

In this section we show how to find a good local fractional matching for weighted graphs. We will not use this result in our algorithm for approximating the maximum weighted matching in Section 3 since a better approximation can be achieved using other ideas combined with the fractional matching proposed for the unweighted case. However, we think the structural result is interesting and could be useful in other computational models.

Define $\mathbf{y} \in \mathbb{R}^E$ where for $e = \{u, v\} \in E$, we set

$$y_e = \min\left(\frac{1}{\deg^e(u) \cdot H(\deg(u))}, \frac{1}{\deg^e(v) \cdot H(\deg(v))}, \frac{1}{\alpha+1}\right)$$

where $\deg^e(u)$ and $\deg^e(v)$ are the number of edges at least as heavy as e that are incident to u and v respectively and $H(r) = 1/1 + 1/2 + \ldots + 1/r$ is the harmonic function.

The next two theorems show that $\mathbf{y}$ is a local fractional matching and

$$\frac{1}{H(D)\cdot(\alpha+1)} \operatorname{match}(G) \leq \operatorname{score}(\mathbf{y}) \leq \frac{\alpha+2}{\alpha+1} \operatorname{match}(G)$$

where $\operatorname{score}(\mathbf{y}) = \sum_e w_e y_e$ and D is the maximum degree of the graph. Note that D can be as large as $n-1$ even for a low arboricity graph. However, since the average degree of G is at most 2α, we expect D to typically be much smaller for many low arboricity graphs of interest.

▶ **Theorem 7.** $\mathbf{y} \in \mathsf{FM}$ *and*

$$\frac{\operatorname{score}(\mathbf{y})}{\operatorname{match}(G)} \leq \begin{cases} \frac{\alpha+2}{\alpha+1} & \text{if } \alpha \text{ odd,} \\ \frac{\alpha+3}{\alpha+2} & \text{if } \alpha \text{ even.} \end{cases}$$

Furthermore, if G is bipartite then $\operatorname{score}(\mathbf{y}) \leq \operatorname{match}(G)$.

Proof. For all $u \in V$,

$$\sum_{e \in E: u \in e} x_e \leq \frac{1}{H(\deg(u))} \sum_{e \in E: u \in e} \frac{1}{\deg^e(u)} \leq \frac{1}{H(\deg(u))}(1/1 + 1/2 + \ldots + 1/\deg(u)) = 1 \ ,$$

and hence $\mathbf{y} \in \mathsf{FM}$. The result of the proof follows as in the proof of Theorem 5 since $y_e \leq 1/(\alpha+1)$ for all e. ◀

▶ **Theorem 8.** $\operatorname{match}(G) \leq H(D)(\alpha+1)\operatorname{score}(\mathbf{y})$ *where D is the maximum degree.*

Proof. Let z_e be the optimum weighted integral matching. Let $0 < w_1 < w_2 < w_3 < \ldots$ be the distinct weights in the graph and let $w_0 = 0$. Let G_k be the unweighted graph formed from the original weighted graph where all edges whose weight is $< w_k$ are deleted and the other edges are given weight 1. Let z_e^k be the optimum unweighted integral matching for G_k and let $\deg_k(u)$ be the degree of node u in G_k.

Then,

$$\begin{aligned} \operatorname{score}(\mathbf{z}) = \sum_e z_e w_e &\leq \sum_k (w_k - w_{k-1}) \sum_{e \in G_k} z_e^k \\ &\leq (\alpha+1)\sum_k (w_k - w_{k-1}) \sum_{e \in G_k} \min\left(\frac{1}{\deg_k(u)}, \frac{1}{\deg_k(v)}, \frac{1}{\alpha+1}\right) \end{aligned}$$

where the last inequality follows by our result for the unweighted case.

But for any $e \in E$,

$$\begin{aligned}
& \sum_{k:e\in G_k} (w_k - w_{k-1}) \min\left(\frac{1}{\deg_k(u)}, \frac{1}{\deg_k(v)}, \frac{1}{\alpha+1}\right) \\
\leq\ & \sum_{k:e\in G_k} (w_k - w_{k-1}) \min\left(\frac{1}{\deg^e(u)}, \frac{1}{\deg^e(v)}, \frac{1}{\alpha+1}\right) \\
\leq\ & w_e \min\left(\frac{1}{\deg^e(u)}, \frac{1}{\deg^e(v)}, \frac{1}{\alpha+1}\right) \\
\leq\ & H(D) w_e y_e
\end{aligned}$$

where the first inequality follows because $\deg_k(u) \geq \deg^e(u)$ for all k such that $e \in G_k$. Therefore $\text{match}(G) \leq H(D)(\alpha+1)\,\text{score}(\mathbf{y})$ as claimed. ◀

2.4 Exact Degree Distribution

Using ideas from the previous sections, we now show that the size of the maximum matching can be approximated up to a $O(\alpha^2)$ factor given just the degree distribution of G. Specifically, consider the following estimate:

$$\tilde{M} = \sum_{u\in V} \min(\alpha + 1 - \deg(u)/2, \deg(u)/2)\,.$$

The next theorem shows that $\tilde{M}$ is a $O(\alpha^2)$ approximation for $\text{match}(G)$.

▶ **Theorem 9.** $\text{match}(G) \leq \tilde{M} \leq \frac{(\alpha+2)^2}{2} \cdot \text{match}(G)$.

Proof. Let h be the number of "heavy" nodes with degree at least $\alpha+2$. Partition the edges E into E_0, E_1, and E_2 depending on whether the edge has zero, one, or two heavy endpoints. Note that E_0 is just the set of shallow edges. Then,

$$\begin{aligned}
& \sum_{u\in V} \min(\alpha + 1 - \deg(u)/2, \deg(u)/2) \\
=\ & \sum_{u\in V} \deg(u)/2 - \max(\deg(u) - \alpha - 1, 0) \\
=\ & |E_0| + |E_1| + |E_2| - \left(\sum_{u:\deg(u)\geq\alpha+2} \max(\deg(u) - \alpha - 1, 0)\right) \\
=\ & |E_0| + |E_1| + |E_2| - \left(\sum_{u:\deg(u)\geq\alpha+2} \deg(u)\right) + h(\alpha+1) \\
=\ & |E_0| + |E_1| + |E_2| - |E_1| - 2|E_2| + h(\alpha+1) \\
=\ & |E_0| - |E_2| + h(\alpha+1)
\end{aligned}$$

First note that $|E_2| \leq \alpha h$ because the number of edges in any induced subgraph is at most α times the number of nodes in that subgraph. Hence,

$$|E_0| - |E_2| + h(\alpha+1) \geq |E_0| + h \geq \text{match}(G)\,.$$

By appealing to Theorem 6 and Theorem 5

$$\begin{aligned}|E_0| - |E_2| + h(\alpha+1) &\le |E_0| + h(\alpha+1)\\ &\le \frac{(\alpha+2)(\alpha+1)}{2}\cdot(|E_0|/(\alpha+1) + 2h/(\alpha+2))\\ &\le \frac{(\alpha+2)(\alpha+1)}{2}\cdot\frac{\alpha+2}{\alpha+1}\cdot \mathrm{match}(G)\\ &\le \frac{(\alpha+2)^2}{2}\cdot\mathrm{match}(G)\,.\end{aligned}$$ ◀

3 Data Stream Algorithms

In this section we briefly discuss the improved algorithmic results that can be achieved via the results from the previous section.

3.1 Arbitrary Order Graph Streams

In the arbitrary order graph stream model, the stream consists of the edges of the input graph G in arbitrary order. The goal is to estimate the size of the maximum cardinality matching using only a single pass over this stream and limited memory.

From Theorem 2, we know we can estimate the size of the maximum cardinality via the following quantity,

$$A := \sum_{\{u,v\}\in E} \min\left(\frac{1}{\deg(u)}, \frac{1}{\deg(v)}, \frac{1}{\alpha+1}\right)\,.$$

To do this we first show that A can be estimated via the quantity,

$$A_S := \sum_{\{u,v\}\in E: u,v\in S} \min\left(\frac{1}{\deg(u)}, \frac{1}{\deg(v)}, \frac{1}{\alpha+1}\right)\,.$$

where S is a subset of V formed by sampling each node independently with probability p. The next lemma shows that A_S is within a $1+\epsilon$ factor of Ap^2 with probability at least $3/4$ assuming p is sufficiently large. Note that a similar approach is taken in Esfandiari et al. [12] and Chitnis et al. [5] in the context of their algorithm to estimate the number of high degree vertices and edges that are not incident to high degree vertices.

▶ **Lemma 10.** *If $p \ge \sqrt{12\epsilon^{-2}A^{-1}}$, then $\mathbb{P}\left[|A_S - Ap^2| \le \epsilon\cdot Ap^2\right] \ge 3/4$.*

Proof. For each edge $e = \{u,v\} \in E$, let $x_e = \min(1/\deg(u), 1/\deg(v), 1/(\alpha+1))$ and define a random variable X_e where $X_e = x_e$ if $u,v\in S$ and $X_e = 0$ otherwise. Note that $A_S = \sum_{e\in E} X_e$. Then, the expectation and variance of A_S are $\mathbb{E}[A_S] = Ap^2$ and

$$\mathbb{V}[A_S] = \sum_{e\in E}\sum_{e'\in E} \mathbb{E}[X_e X_{e'}] - \mathbb{E}[X_e]\,\mathbb{E}[X_{e'}]\,.$$

Note that

$$\sum_{e'\in E}\mathbb{E}[X_eX_{e'}] - \mathbb{E}[X_e]\,\mathbb{E}[X_{e'}] = \begin{cases} x_e^2(p^2-p^4) & \text{if } e = e' \\ x_e x_{e'}(p^3-p^4) & \text{if } e \text{ and } e' \text{ share exactly one endpoint}\,. \\ 0 & \text{if } e \text{ and } e' \text{ share no endpoints}\end{cases}$$

Since the sum of all $x_{e'}$ that share an endpoint with e is at most 2 because $\mathbf{x} \in \mathsf{FM}$,

$$\mathbb{V}[A_S] \le \left(\sum_{e\in E} x_e^2(p^2 - p^4)\right) + 2A(p^3 - p^4) \le 3Ap^2 .$$

We then use Chebyshev's inequality to obtain

$$\mathbb{P}\left[|A_S - Ap^2| \le \epsilon Ap^2\right] \le \frac{3Ap^2}{\epsilon^2 A^2 p^4} = \frac{3}{\epsilon^2 Ap^2} \le 3/4 . \qquad ◀$$

Given this key lemma, the algorithm and analysis proceed similarly to that of Esfandiari et al. [12]. Specifically, two algorithms are run in parallel: a greedy matching algorithm and a sampling-based algorithm. The greedy matching algorithm uses $O(n^{2/3}\log n)$ space to find a maximal matching of size at least $\min(n^{2/3}, \mathrm{match}(G)/2)$. The sampling-based algorithm uses $O(\alpha n^{2/3}\log n)$ space to sample each node with probability $p = \Theta(\epsilon^{-1}/n^{2/3})$ and then find all edges whose endpoints are both sampled along with the degrees of the sampled edges. If the greedy matching has size less than $n^{2/3}$ then it is necessarily a 2 approximation of $\mathrm{match}(G)$. If not, we can use the estimate of A based on the nodes sampled since in this case $A = \Omega(n^{2/3})$. Similarly, extensions of the above approach for dynamic graph streams [5, 4] go through with the improved approximation factor. To summarize:

▶ **Theorem 11.** *There exists a single pass data stream algorithm using $O(\alpha\epsilon^{-1}n^r \log \delta^{-1})$ space that returns a $(\alpha + 2)(1 + \epsilon)$ approximation of the maximum matching with probability at least $1 - \delta$. In the insert-only model, $r = 2/3$ and in the insert-delete model $r = 4/5$.*

3.2 Adjacency List Graph Streams

In the *adjacency list model*[1] the edges incident to each node v appear consecutively in the stream [23, 3, 2]. Thus, every edge $\{u, v\}$ will appear twice: once when we view the adjacency list of u and once for v. Aside from that constraint, the stream is ordered arbitrarily. For example, for the graph consisting of a cycle on three nodes $V = \{v_1, v_2, v_3\}$, a possible ordering of the stream could be $\langle v_3v_1, v_3v_2, v_2v_3, v_2v_1, v_1v_2, v_1v_3\rangle$. Note that in this model it is trivial to compute

$$\tilde{M} = \sum_{u\in V} \min(\alpha + 1 - \deg(u)/2, \deg(u)/2) .$$

in $O(\log n)$ space since the degree of a node can be calculated exactly when the adjacency list of that node appears. The next theorem immediately follows from Theorem 9.

▶ **Theorem 12.** *An $(\alpha + 2)^2/2$-approximation of the size of maximum matching can be computed using $O(\log n)$ in the adjacency list model. In particular, this yields a 12.5-approximation for planar graphs.*

3.3 Extension to Weighted Graphs

Let $G = (V, E)$ be a weighted graph where edge e has weight $w_e \in [1, W]$ where $W = \mathrm{poly}(n)$. In this section we show that it is possible to reduce the problem of finding a large weighted matching in G to finding large cardinality matchings. Specifically, we show that given a

[1] The adjacency list order model is closely related to the vertex arrival model [15, 18] and row-order arrival model considered in the context of linear algebra problems [6, 14].

t-approximation algorithm for the unweighted problem, there is a $2(1+\epsilon)t$-approximation the maximum weighted problem where the latter algorithm using a factor $O(\epsilon^{-1}\log n)$ more space. This reduction uses ideas from work by Crouch and Stubbs [7].[2] This immediately implies a $2(2+\alpha)(1+\epsilon)$-approximation algorithm for weighted graphs in the arbitrary order model and a $(2+\alpha)^2(1+\epsilon)$-approximation algorithm for weighted graphs in the adjacency list model.

Reduction to Unweighted Matchings

For $k = 0, 1, \ldots, \lfloor \log_{1+\epsilon} W \rfloor$, define the unweighted graph $G_k = (V, E_k)$ where $e \in E_k$ iff $w_e \geq (1+\epsilon)^k$ where w_e is the weight of e in the original weighted graph. Note that $E_0 \subseteq E_1 \subseteq E_2 \subseteq \ldots$ and, in particular, $E_0, E_1, \ldots$ is *not* a partition of E.

▶ **Lemma 13.** *Let* match(G) *be the weight of the maximum weighted matching in G and let $\tilde{m}_k$ be a t-approximation of the size of the maximum cardinality matching in G_k. Then,*

$$\text{match}(G)/t \leq \sum_{k\geq 0} f(k)\cdot \tilde{m}_k \leq 2\cdot(1+\epsilon)\cdot \text{match}(G)$$

where

$$f(k) = \begin{cases} (1+\epsilon)^{k+1} - (1+\epsilon)^k & \textit{if } k > 0\,, \\ (1+\epsilon) & \textit{if } k = 0\,. \end{cases}$$

Proof. Let m_k be the size of the maximum cardinality matching in G_k and let M be the set of edges in the maximum weighted matching in G. To prove the left inequality, observe that

$$\sum_{k\geq 0} f(k)\cdot \tilde{m}_k \geq \sum_{k\geq 0} f(k)\cdot m_k/t \geq \sum_{k\geq 0} f(k)\cdot |M\cap E_k|/t \geq \text{match}(G)/t\ ,$$

where the last inequality follows since

$$(1+\epsilon)w_e \geq \sum_{k:w_e\geq(1+\epsilon)^k} f(k) \geq w_e\,. \tag{1}$$

We now prove the right inequality. Consider the matching R formed by taking a maximal matching in E_r where $r = \lfloor \log_{1+\epsilon} W\rfloor$; extending this to a maximal matching in E_{r-1}; extending this to a maximal matching in E_{r-2} as so on. Note that since $R\cap E_k$ is a maximal matching in E_k, we have $\tilde{m}_k \leq m_k \leq 2|R\cap E_k|$. Therefore,

$$\sum_{k\geq 0} f(k)\cdot \tilde{m}_k \leq 2\sum_{k\geq 0} f(k)\cdot |R\cap E_k| \leq 2(1+\epsilon)\sum_{e\in R} w_e \leq 2(1+\epsilon)\,\text{match}(G)\ ,$$

where the second last inequality follows from Eq. 1. ◀

4 Conclusion

We established better approximation ratios for the data stream problem of estimating the maximum weight and cardinality matchings in graphs of bounded arboricity α. The main technical result is that the relatively simple quantity $\sum_{\{u,v\}\in E} \min(1/\deg(u), 1/\deg(v), 1/(\alpha+1))$ determines the size of the maximum cardinality matching up to a factor of $(\alpha+2)$, e.g., 5 in the case of planar graphs, and this quantity can be estimated efficiently in the data stream model. Other results included establishing that the degree distribution determines the size of the maximum cardinality matching up to a factor of $(\alpha+2)^2/2$, e.g., 12.5 in the case of planar graphs.

[2] Concurrent with our work, Grigorescu, Monemizadeh, and Zhou [16] designed a similar reduction.

References

1 Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, 2013. doi:10.1016/j.ic.2012.10.006.

2 Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proc. of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA.*, pages 623–632, 2002. URL: http://dl.acm.org/citation.cfm?id=545381.545464.

3 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 SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 253–262, 2006. doi:10.1145/1142351.1142388.

4 Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Algorithms – ESA 2015 – 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 263–274, 2015. doi:10.1007/978-3-662-48350-3_23.

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 *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1326–1344, 2016. doi:10.1137/1.9781611974331.ch92.

6 Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 205–214, 2009. doi:10.1145/1536414.1536445.

7 Michael Crouch and Daniel S. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 96–104, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.96.

8 Michael S. Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic graphs in the sliding-window model. In *Algorithms – ESA 2013 – 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 337–348, 2013. doi:10.1007/978-3-642-40450-4_29.

9 Andrzej Czygrinow, Michal Hanckowiak, and Edyta Szymanska. Fast distributed approximation algorithm for the maximum matching problem in bounded arboricity graphs. In *Algorithms and Computation, 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings*, pages 668–678, 2009. doi:10.1007/978-3-642-10631-6_68.

10 Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards, 69:125-130*, 1965.

11 Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011. doi:10.1137/100801901.

12 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 *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, 2015. doi:10.1137/1.9781611973730.81.

13 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2):207–216, 2005. doi:10.1016/j.tcs.2005.09.013.

14 Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P. Woodruff. Frequent directions: Simple and deterministic matrix sketching. *CoRR*, abs/1501.01711, 2015. URL: http://arxiv.org/abs/1501.01711.

15 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485, 2012. URL: http://portal.acm.org/citation.cfm?id=2095157&CFID=63838676&CFTOKEN=79617016.

16 Elena Grigorescu, Morteza Monemizadeh, and Samson Zhou. Estimating weighted matchings in o(n) space. *CoRR*, abs/1604.07467, 2016. URL: http://arxiv.org/abs/1604.07467.

17 Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. In *Proc. of the 28th Conference on Computational Complexity, CCC 2013, Palo Alto, California, USA, 5-7 June, 2013*, pages 287–298, 2013. doi:10.1109/CCC.2013.37.

18 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.

19 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751, 2014. doi:10.1137/1.9781611973402.55.

20 Christian Konrad. Maximum matching in turnstile streams. In *Algorithms – ESA 2015 – 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 840–852, 2015. doi:10.1007/978-3-662-48350-3_70.

21 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. doi:10.1007/978-3-642-32512-0_20.

22 Christian Konrad and Adi Rosén. Approximating semi-matchings in streaming and in two-party communication. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 637–649, 2013. doi:10.1007/978-3-642-39206-1_54.

23 Madhusudan Manjunath, Kurt Mehlhorn, Konstantinos Panagiotou, and He Sun. Approximate counting of cycles in streams. In *Algorithms – ESA 2011 – 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, pages 677–688, 2011. doi:10.1007/978-3-642-23719-5_57.

24 Andrew McGregor. Finding graph matchings in data streams. *APPROX-RANDOM*, pages 170–181, 2005.

25 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014. doi:10.1145/2627692.2627694.

26 Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012. doi:10.1007/s00453-010-9438-5.

A Robust and Optimal Online Algorithm for Minimum Metric Bipartite Matching

Sharath Raghvendra[*]

Dept. of Computer Science, Virginia Tech, Blacksburg, USA
sharathr@vt.edu

Abstract

We study the Online Minimum Metric Bipartite Matching Problem. In this problem, we are given point sets S and R which correspond to the server and request locations; here $|S| = |R| = n$. All these locations are points from some metric space and the cost of matching a server to a request is given by the distance between their locations in this space. In this problem, the request points arrive one at a time. When a request arrives, we must immediately and irrevocably match it to a "free" server. The matching obtained after all the requests are processed is the *online* matching M. The cost of M is the sum of the cost of its edges. The performance of any online algorithm is the worst-case ratio of the cost of its online solution M to the minimum-cost matching.

We present a deterministic online algorithm for this problem. Our algorithm is the first to simultaneously achieve optimal performances in the well-known adversarial and the random arrival models. For the adversarial model, we obtain a competitive ratio of $2n - 1 + o(1)$; it is known that no deterministic algorithm can do better than $2n - 1$. In the random arrival model, our algorithm obtains a competitive ratio of $2\mathrm{H}_n - 1 + o(1)$; where H_n is the nth Harmonic number. We also prove that any online algorithm will have a competitive ratio of at least $2\mathrm{H}_n - 1 - o(1)$ in this model.

We use a new variation of the offline primal-dual method for computing minimum cost matching to compute the online matching. Our primal-dual method is based on a relaxed linear-program. Under metric costs, this specific relaxation helps us relate the cost of the online matching with the offline matching leading to its robust properties.

1998 ACM Subject Classification G.2.2 [Graph Theory] Graph Algorithms

Keywords and phrases Online Algorithms, Metric Bipartite Matching

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.18

1 Introduction

In an era of instant gratification, consumers desire speedy access to goods and services. Several new business ventures promise on-demand delivery of such services to consumers. Typically, these ventures have servers in various locations of the city and when a new request arrives, they match one of the available servers to this request. The cost associated with this match is often a metric cost; for instance, it could be the minimum distance traveled by the server to reach the request. A primary objective is to minimize the overall cost of the assignments made. This problem is difficult because all of the request locations are not known in advance.

Each server may have a maximum capacity of how many requests it can serve. A central problem in online algorithms is the *k-server* problem where each of the k servers has a

[*] The author would like to acknowledge the support of NSF CRII grant CCF-1464276 in conducting this research.

capacity to serve an arbitrary number of requests. In the case where the server capacity is 1, the problem reduces to the well-studied *online minimum metric bipartite matching* problem.

To define this problem, let S be a set of servers and let R be a set of requests. We assume that the locations in $S \cup R$ are points from some metric space. Let $\mathtt{d}(a, b)$ be the distance between any two points in this space. Consider a complete bipartite graph $G(S \cup R, S \times R)$, $|S| = |R| = n$, with the edge set $S \times R$. Every pair of server and request, $(s, r) \in S \times R$ has a distance $\mathtt{d}(s, r)$. We refer to this distance as the *cost* of server s serving request r. A *matching* $M \subseteq S \times R$ is any set of vertex-disjoint edges of the bipartite graph $G(S \cup R, S \times R)$. The cost of any matching M is given by $w(M) = \sum_{(s,r)\in M} \mathtt{d}(s, r)$. A *perfect matching* is a matching where every server in S is serving exactly one request in R, i.e., $|M| = n$. A *minimum-cost perfect matching* is a perfect matching with the minimum cost.

Ideally, we would like to match servers to requests so that the cost of this matching is as small as possible. However, in the *online metric bipartite matching problem*, the requests arrive one at a time and when any request arrives, we have to immediately and irrevocably match it to some unmatched server. The resulting matching is referred to as an *online matching*. Designing an online algorithm which finds a matching with minimum-cost is impossible because, for any partial assignment made by the algorithm, an adversary can easily fill up the remaining request locations in R so that this partial assignment becomes sub-optimal. Therefore, we want our algorithm to compute an online matching which is only near optimal. For any input S, R and any arrival order of requests in R, we say our algorithm is *α-competitive*, for $\alpha > 1$, when the cost of the online matching M is at most α times the minimum cost, i.e.,

$$w(M) \leq \alpha w(M_{\text{OPT}}).$$

Here M_{OPT} is the minimum-cost matching of the locations in S and R.

In the above discussion, note the role of the adversary. In the *adversarial model*, the adversary knows the server locations and the assignments made by the algorithm and generates a sequence to maximize α. In order to account for adversarial input sequence, algorithms which work well in this model may become very cautious in making low-cost assignments. As a result, their performance may be hampered on realistic input sequences.

A less pessimistic model is the *random arrival model*. In this model [13], the adversary chooses the set of request locations R at the start but the arrival order is a permutation chosen uniformly at random from the set of all possible permutations; we refer to this as a *random permutation*. For any input S, R and an arrival order which is a random permutation of R, we say our algorithm is *α-competitive*, for $\alpha > 1$, when the expected cost of the online matching M is at most α times the minimum cost, i.e.,

$$\mathbb{E}[w(M)] \leq \alpha w(M_{\text{OPT}}).$$

In practical situations, one can assume that the requests locations are independent and identically distributed (i.i.d.) random variables from an unknown but fixed distribution $\mathscr{D}$. On many occasions, using historical data, one can learn this distribution $\mathscr{D}$. These KnownIID and UnknownIID models are weaker than the random arrival model. Therefore, the competitive ratio of an algorithm in the random arrival model is an upper bound on its performance in the KnownIID and the UnknownIID models; see [9] for algorithms in these models.

Another popular model of theoretical interest is the *oblivious adversary* model. In this model, the adversary knows the algorithm and decides the request locations and their arrival order. However, the online algorithm is a randomized algorithm and the adversary does not know the random choices made by the algorithm. This model is weaker than the adversarial model but stronger than the random arrival model.

Below is the summary of relative hardness of all these models

$$\text{Adversarial} \succ \text{Oblivious Adversary} \succ \text{Random Arrival} \succ \textsc{UnknownIID} \succ \textsc{KnownIID}.$$

Existing Work. Solutions for the k-server problem and the online bipartite matching problem use similar mathematical tools and methodologies. Both of these problems have been extensively studied in the adversarial model and the oblivious model. However, we are not aware of any work on these problems under the random arrival model.

The k-server problem is central to the theory of online algorithms. The problem was first posted by Manasse *et al.* [14]. In the adversarial model, the best-known deterministic algorithm for this problem is the $2k-1$-competitive work function algorithm [12]. In this problem, we assume there are k servers, each of which can serve arbitrary many of the n arriving requests. It is known that no deterministic algorithm can achieve a competitive ratio better than k and is conjectured that in fact there is a k-competitive algorithm for this problem. This conjecture is popularly called the *k-server conjecture*.

For the online metric bipartite matching problem, in the adversarial model, there is a $2n-1$-competitive deterministic algorithm by Khuller *et al.* [10] and Kalyanasundaram and Pruhs [8]. They also show that no online algorithm can achieve a better competitive ratio in this model.

For the oblivious adversary, there are $O(\text{poly} \log n)$-competitive algorithms for both the k-server problem and the online metric bipartite matching problem. Bansal *et al.* [5] achieve an $O(\log^2 n)$-competitive algorithm for the metric bipartite matching problem. For the k-server problem, Bansal *et al.* [4] presented a $O(\text{poly} \log n \log k)$-competitive algorithm.

All of these algorithms use a standard approach. They first embed the metric space into a tree metric that leads to a $O(\log n)$ distortion in costs. Then, they design a $\log n$-competitive algorithm for this tree metric. As a consequence, these results obtain a $\log^2 n$-competitive algorithm ($\text{poly} \log n$-competitive for the k-server). The bottleneck in improving existing work is the $O(\log n)$-distortion associated with the tree metric. An open question is whether one can design an $O(\log n)$-competitive algorithm for these problems.

Also note that for bounded doubling dimension metric, there is an a $O(d \log n)$-competitive algorithm in the oblivious model [6]; here d is the doubling dimension of the metric space. In the adversarial model, the question of finding a deterministic $O(1)$-competitive online algorithm for the line metric remains an important open question; see [3, 11] for results on this special case. We would like to note the existence of several fast primal-dual algorithms to compute approximate (offline) matching in metric and geometric settings [7, 2, 16, 17, 1].

Our Results. In this paper, we give a robust deterministic online algorithm for the metric bipartite matching problem. Our algorithm achieves an optimal performance of $2n-1$ in the adversarial model. This same algorithm has an exponentially better performance in the random arrival model where we obtain optimal $2\mathrm{H}_n - 1 + o(1)$-competitive ratio. Here H_n is the nth harmonic number (approximately $\ln n$). We also prove that no algorithm can achieve a competitive ratio better than $2\mathrm{H}_n - 1 - o(1)$.

To our knowledge, this is the first online algorithm which achieves optimal performances in two different models simultaneously. Our algorithm's robustness across different models of adversaries is also crucial in practical settings where there is limited information about the model of adversary.

Unlike previous work, our approach does not use a tree metric, thereby allowing us to achieve a $O(\log n)$-competitive algorithm in the random arrival model.

Our Techniques. Many deterministic online algorithms use the best minimum-cost offline solution to construct the online solution. This includes the work-function algorithm for the k-server problem and the deterministic algorithms for the online minimum metric bipartite matching problem. Our approach is similar in style, except we use, for some $t > 1$, a t-approximate minimum-cost solution to guide our online solution. This approximate matching is derived from a relaxed linear program where the constraint for every non-matching edge is relaxed by a multiplicative factor of t. We will fix the value of t for the entire execution of the algorithm. When a new request arrives, our algorithm updates the offline matching by computing an augmenting path P. For the online matching, the algorithm simply matches the two free end points of P.

We observe that for larger values of t, the algorithm picks an augmenting path P of a smaller length leading to lower cost of the online matching. On the other hand, larger values of t causes the offline matching to be a weaker approximation which leads to a weaker bound for the online matching. Therefore, it may seem that the best trade-off between these opposing observations is achieved at some finite value of t.

However, surprisingly, we show that the performance of our online algorithm improves as $t \to \infty$. We show that the competitive ratio of our algorithm is $(2 + \frac{2}{t-1})n - (1 + \frac{2}{t-1})$ in the adversarial model and $(2 + \frac{2}{t-1})\mathrm{H}_n - (1 + \frac{2}{t-1})$ in the random arrival model.

2 Preliminaries

In this section, we present preliminary notations required to describe our algorithm.

Given a matching M^* on this bipartite graph, an *alternating path* (or cycle) is a simple path (resp. cycle) whose edges alternate between those in M^* and those not in M^*. We refer to any vertex that is not matched in M^* as a *free vertex.* An *augmenting path* P is an alternating path between two free vertices. We can *augment* M^* by one edge along P if we remove the edges of $P \cap M^*$ from M^* and add the edges of $P \setminus M^*$ to M^*. After augmenting, the new matching is given by $M^* \leftarrow M^* \oplus P$, where $\oplus$ is the symmetric difference operator. For a parameter $t \geq 1$, we define the *t-net-cost* of an augmenting path P as follows:

$$\phi_t(P) = t\left(\sum_{(s,r)\in P\setminus M^*} \mathtt{d}(s,r)\right) - \sum_{(s,r)\in P\cap M^*} \mathtt{d}(s,r)\,.$$

When $t = 1$, we can interpret the t-net-cost of a path as the increase in the cost of the matching due to augmenting it along P, i.e., for $t = 1$, $\phi_1(P) = w(M \oplus P) - w(M)$. The well-known Hungarian method iteratively augments along an augmenting path with the minimum 1-net-cost to compute the optimal matching. For $t > 1$, the t-net-cost $\phi_t(P)$ can be very different from $w(M \oplus P) - w(M)$. As noted earlier, larger values of t yields smaller length augmenting paths. In Figure 1, there are two augmenting paths. Let the augmenting path from r to s' be P' and the augmenting path from r to s be P. When $t = 1$, the $\phi_1(P) = \phi_1(P') = 1$. However, $\phi_2(P') = 3$ and $\phi_2(P) = 4$. As t increases, the difference between the t-net-costs of these paths is magnified.

We derive an alternate interpretation of the t-net-cost in Section 3 which will be crucial in providing guarantees for our online solution. The definition of t-net-cost easily extends to alternating paths and cycles as well.

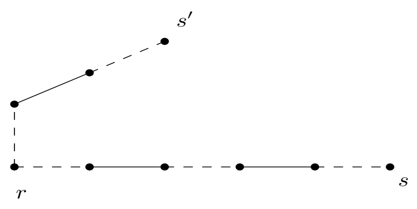


Figure 1 All solid edges are edges in the matching M^* and the dashed edges are not in M^*. The cost of every edge is 1.

Feasibility of a Matching. For every vertex v of the graph $G(S \cup R, S \times R)$, let its dual weight be $y(v)$. For a parameter $t \geq 1$, we define a *t-feasible* matching to be a matching M^* and a set of dual weights $y(\cdot)$ on the vertex set such that for every edge between request $r \in R$ and server $s \in S$, we have

$$y(s) + y(r) \leq t\mathtt{d}(s, r), \quad (1)$$
$$y(s) + y(r) = \mathtt{d}(s, r) \quad \text{for } (s, r) \in M^*. \quad (2)$$

Using t-feasibility and t-net-cost we describe our algorithm next.

2.1 Algorithm

In this section, we present our algorithm without fixing the parameter t. Eventually, to obtain the bounds on the competitive ratio, we set $t = n^2 + 1$.

For every vertex $v \in S \cup R$, our algorithm will maintain a dual weight $y(v)$. At the start of the algorithm, the dual weight of every vertex is set to 0. Recollect that, in the online setting, requests from the set R arrive one at a time. For those requests $r' \in R$ which have not yet arrived, their dual weight remains 0, i.e., $y(r') = 0$. The algorithm also maintains two matchings M and M^*; both these matchings are initialized to $\emptyset$ at the start. M and M^* match all the request seen so far to servers in S. The matching M^* together with the dual weights $y(\cdot)$ is a t-feasible matching; we refer to this as the *offline matching*. The matching M, on the other hand, is the *online matching*.

Algorithm. Given a new request r, our algorithm computes the minimum t-net-cost augmenting path P with respect to matching M^*. P starts at r and ends at some free vertex s. The algorithm updates M^* by augmenting it along P, i.e., $M^* \leftarrow M^* \oplus P$. For the online matching M, the algorithm will match the server s to r, i.e., $M \leftarrow M \cup \{(s, r)\}$.

At any given stage in the algorithm, let S_F be the set of free servers in S with respect to the offline matching M^*. It follows from the description of our algorithm that S_F is also the set of free servers with respect to the online matching M.

Our algorithm maintains the following invariants:

(I1) M^* and dual weights $y(\cdot)$ form a t-feasible matching, and,

(I2) for every vertex $s \in S$, $y(s) \leq 0$ and if $s \in S_F$, $y(s) = 0$.

Next, we present an $O(n^2)$ time algorithm to compute the minimum t-net-cost augmenting path P and update the matchings M, M^* and the dual weights. After describing the algorithm, we prove the invariants (I1) and (I2).

Algorithm Details. To compute a minimum t-net-cost augmenting path P with respect to the offline matching M^*, we construct a *weighted residual graph* G_{M^*} with $S \cup R$ as the vertex set and E_{M^*} as the set of edges as follows. Let $\overrightarrow{sr}$ represent an edge directed from s to r. For every edge $(s, r) \in M^*$, we have an edge $\overrightarrow{sr}$ in E_{M^*}. For every edge $(s, r) \in (S \times R) \setminus M^*$, there is an edge $\overrightarrow{rs}$ in E_{M^*}. Every edge $\overrightarrow{ab} \in E_{M^*}$ is assigned a cost as follows:

- If $(a, b) \in M^*$, we set the cost of the edge to be the slack $s(a, b) = \mathtt{d}(a, b) - y(a) - y(b)$. From t-feasibility (condition (2)) of M^*, we know the slack of every edge in the matching is $s(a, b) = 0$.
- If $(a, b) \notin M^*$, we set the cost of the edge (a, b) to be the slack $s(a, b) = t\mathtt{d}(a, b) - y(a) - y(b)$. From t-feasibility (condition (1)) of M^*, we know $s(a, b) \geq 0$.

By construction, every edge in E^*_M has a non-negative edge cost. Also, notice that the set of nodes in $G(S \cup R, S \times R)$ (henceforth referred to as G) and G_{M^*} are identical. G_{M^*} and G have the same set of edges except that the edges of G_{M^*} have directions. For any directed path $\overrightarrow{P}$ in G_{M^*}, we can define its *associated path* P by replacing every edge $\overrightarrow{ab} \in \overrightarrow{P}$ with the corresponding undirected edge (a, b) from G. For any directed path $\overrightarrow{P}$ in G_{M^*}, its associated path P is an alternating path in G. More so, if the two end vertices of $\overrightarrow{P}$ are free vertices, then the associated path P will be an augmenting path.

To compute the minimum t-net-cost augmenting path P, we simply execute Dijkstra's algorithm and find the minimum-cost path from r to every other node in G_{M^*}. For any node v, let d_v be the cost of the shortest path from r to v. Among all free servers of S, we pick $s \in S_F$ with the lowest minimum-cost path from r in G_{M^*}, i.e., $s = \arg\min_{s' \in S_F} d_{s'}$. Let this lowest minimum-cost path be $\overrightarrow{P}$. Clearly $\overrightarrow{P}$ is a directed path from r to s. Let P be the associated augmenting path of $\overrightarrow{P}$ in G. In Lemma 1 and Corollary 2, we show that P is the minimum t-net-cost augmenting path starting at r.

Before augmenting the matching M^* along P, we update the dual weights of all nodes of $S \cup R$. Let $d_s = d$ be the cost of the directed path $\overrightarrow{P}$. We update the dual weight of every node v as follows:

(a) If $d_v \geq d$, then $y(v)$ remains unchanged.
(b) If $d_v < d$, and $v \in R$, then we increase the dual weight $y(v) \leftarrow y(v) + d - d_v$
(c) If $d_v < d$, and $v \in S$, then we decrease the dual weight $y(v) \leftarrow y(v) - d + d_v$.

In Lemma 4, we show that the updated dual weights and the matching M^* are t-feasible. We also show, in Lemma 4, that after the dual updates, every edge in $P \setminus M^*$ will satisfy (1) with equality.

At this point, we update matching M^* by augmenting M^* along P, i.e., $M^* \leftarrow M^* \oplus P$. We also update the dual weight of every vertex $r' \in R \cap P$ as follows: $y(r') \leftarrow y(r') - (t-1)\mathtt{d}(s', r')$; here s' is the match of r' in the updated M^*. Lemma 5 will show that the matching after the augmentation and the updated dual weights remain t-feasible.

In processing a new request, note that we update the dual weights twice. First, we update them right before augmenting the matching along P as describe in a–c. Then, immediately after augmenting M^* along the path P, we update the dual weight again. In Lemma 4 and Lemma 5, we will show that both these updates do not violate the t-feasibility property of M^*. Therefore, (I1) holds. The proofs of Lemma 1, Lemma 3, Lemma 4, Lemma 5 and Proof of (I2) are variants of the proofs for the standard primal-dual based Hungarian method. For the sake of completion, we present these proofs. An expert may choose to skip these proofs.

The following lemma will show that Dijkstra's algorithm will compute the minimum t-net-cost path between r and any free server s.

▶ **Lemma 1.** *For any free server $s \in S_F$, let $\overrightarrow{P}$ be a directed path from the new request r to s in G_{M^*} with a cost d'. Then, the associated path P of $\overrightarrow{P}$ is an augmenting path whose t-net-cost is d'.*

Proof. The cost of $\overrightarrow{P}$ is:

$$d' = \sum_{\overrightarrow{ab} \in \overrightarrow{P}} s(a, b) \tag{3}$$

For the associated path P, from the definition of t-net-cost and the feasibility of dual weights, we have

$$\begin{aligned} \phi_t(P) &= t(\sum_{(s',r') \in P \setminus M^*} \mathtt{d}(s', r')) - \sum_{(s',r') \in P \cap M^*} \mathtt{d}(s', r') \\ &= t(\sum_{(s',r') \in P \setminus M^*} \mathtt{d}(s', r')) - \sum_{(s',r') \in P \cap M^*} (y(s') + y(r')) \end{aligned} \tag{4}$$

Since the first and the last vertex of the associated path P, i.e., s and r, are unmatched in M^*, both the first and the last edge of P is not in the matching M^*. Therefore, we can write (4) as:

$$\phi_t(P) = \sum_{(s',r') \in P \setminus M^*} (t(\mathtt{d}(s', r')) - y(s') - y(r')) + y(s) + y(r) \tag{5}$$

Note that $y(s)$ is 0 by invariant (I2) and by construction $y(r)$ is 0. For every edge $(s', r') \in P \cap M^*$, $\mathtt{d}(s', r') - y(s') - y(r') = 0$. Combining this, we can rewrite equation 5 as

$$\begin{aligned} \phi_t(P) &= \sum_{(s',r') \in P \setminus M^*} (t(\mathtt{d}(s', r')) - y(s') - y(r')) + \sum_{(s',r') \in P \cap M^*} (\mathtt{d}(s', r') - y(s') - y(r')) \\ &= \sum_{\overrightarrow{ab} \in \overrightarrow{P}} s(a, b) = d' \end{aligned}$$ ◀

As an immediate corollary to Lemma 1, we conclude that the path chosen by our algorithm is the minimum t-net-cost path.

▶ **Corollary 2.** *In processing the new request r, the augmenting path P chosen by our algorithm has the smallest t-net-cost among all augmenting paths that begin at r.*

In order to show that M^* and dual weights $y(\cdot)$ form a t-feasible, we need to show that all edges in the complete bipartite graph $G(S \cup R, S \times R)$ satisfy t-feasibility conditions (1) and (2). This includes the edges incident on requests that have not yet arrived. The following simple lemma will show that any such edge will satisfy t-feasibility conditions at all times.

▶ **Lemma 3.** *At any stage of the algorithm, consider any edge $(r', s') \in R \times S$ where r' is a request that has not yet arrived. We claim that the edge (r', s') satisfies the t-feasibility condition.*

Proof. Since r' has not yet arrived, r' is a free node. Therefore, the edge (r', s') is not in the current matching M^*. By construction, every request that has not yet arrived has a dual weight of 0. Therefore, $y(r') = 0$. From invariant (I2), $y(s') \leq 0$. Since $t \geq 1$ and $\mathtt{d}(s', r') \geq 0$, we have

$$y(r') + y(s') \leq 0 \leq t\mathtt{d}(s', r'),$$

showing that (r', s') satisfies (1). ◀

From this point onwards, to show that M^* is t-feasible, we will focus only on the edges incident on requests that have already arrived. Lemma 3, allows us to ignore all other edges incident on requests which have not yet arrived.

The following lemma shows that the updated dual weights before augmenting along P satisfy the desired properties:

▶ **Lemma 4.** *Let $y(\cdot)$ be the dual weights assigned to $S \cup R$ before executing Dijkstra's algorithm and let $y'(\cdot)$ be the updated dual weights computed before augmenting M^* along P. The matching M^* and dual weights $y'(\cdot)$ are t-feasible. Furthermore, suppose P is the augmenting path chosen by the algorithm. Then, every edge of P has 0 slack with respect to the updated dual weights $y'(\cdot)$, i.e., for every $(s', r') \in M^* \cap P$, $\mathtt{d}(s', r') - y(s') - y(r') = 0$ and for every $(s', r') \in P \setminus M^*$, $t\mathtt{d}(s', r') - y(s') - y(r') = 0$.*

Proof. First, for any edge $(s', r') \in M^*$, we show that the updated dual weights does not violate feasibility condition (2). For any edge $(s', r') \in M^*$ there is a directed edge $\overrightarrow{s'r'}$ in G_{M^*}. By construction, all edges incident on r' except for the edge $\overrightarrow{s'r'}$ is directed away from r' (edges that are not in the matching are directed away from the requests in G_{M^*}). Therefore any path in G_{M^*} from the new request r to r' must contain the edge $\overrightarrow{s'r'}$; note that since $(s', r') \in M^*$, it has 0 slack and therefore the cost of this edge in G_{M^*} is $s(s', r') = 0$. Therefore, the shortest path from r to r' has the same cost as the shortest path from r to s', i.e., $d_{s'} = d_{r'}$. If $d_{s'} \geq d$, the dual weights of s' and r' are not updated (update condition (a)), and therefore the edge continues to satisfy (2). On the other hand, if $(d_{s'} = d_{r'}) < d$, the dual weight $y(r')$ increases by $d - d_{r'}$ and the dual weight of $y(s')$ decreases by $d - d_{s'}$. Since $d_{s'} = d_{r'}$, we have

$$y'(s') + y'(r') = y(s') - d + d_{s'} + y(r') + d - d_{r'} = y(s') + y(r') = \mathtt{d}(s', r').$$

Therefore, every edge $(s', r') \in M^*$ continues to satisfy the t-feasibility condition (2).

For any edge $(s', r') \in (S \times R) \setminus M^*$, there is an edge $\overrightarrow{r's'}$ in G_{M^*}. Since shortest path costs satisfy triangle inequality, we have $d_{s'} \leq d_{r'} + s(r', s')$, or

$$d_{s'} - d_{r'} \leq s(r', s').$$

After the dual weights are updated, we have

$$y'(r') + y'(s') = y(r') + d - d_{r'} + y(s') - d + d_{s'} \leq y(s') + y(r') + s(s', r') \leq t\mathtt{d}(s', r').$$

Therefore, every edge (s', r') remains feasible after the dual updates.

For every edge in $P \cap M^*$, we have already shown that the edge satisfies t-feasibility condition (2) and therefore has a slack of 0. Next, we show that the slack on every edge $(s', r') \in P \setminus M$ is also 0. Since $(s', r') \notin M^*$, there is a directed edge $\overrightarrow{r's'}$ in G_{M^*}. From the optimal substructure property of shortest paths, we have $d_{s'} = d_{r'} + s(r', s')$,. Therefore, we have

$$y'(r') + y'(s') = y(r') + d - d_{r'} + y(s') - d + d_{s'} = y(s') + y(r') + s(s', r') = t\mathtt{d}(s', r'),$$

implying that the slack on every such edge after the dual weights are updated is 0. ◀

At this point, the algorithm augments M^* along P and updates the dual weights again. The following lemma shows that the augmentation process and the updated dual weights continue to satisfy t-feasibility conditions.

▶ **Lemma 5.** *After augmentation, the updated dual weights together with the updated matching M^* form a t-feasible matching.*

Proof. For the sake of this proof, let us refer to the matching before augmenting along P as M^* and the matching after augmentation as M', i.e., $M' \leftarrow M^* \oplus P$. Also, let $y(\cdot)$ represent the dual weight right before augmenting M^* along P and let $y'(\cdot)$ represent the updated dual weight after augmentation.

Augmenting along P removed edges from $P \cap M^*$ and adds edges of $P \setminus M^*$ to matching. Then, the algorithm updates dual weights of every vertex in $R \cap P$. Every edge in $(s', r') \in M^* \cap M'$ is vertex-disjoint from the path P. Therefore, for every such edge, the dual weights of s' and r' remains unchanged and the edge continues to satisfy (2).

For any edge $(s', r') \in P \setminus M^*$, we know (s', r') is in the updated matching M'. From Lemma 4, every such edge has a 0 slack and therefore,

$$y(s') + y(r') = t\mathtt{d}(s', r').$$

The updated dual weights for r' is $y'(r') \leftarrow y(r') - (t-1)\mathtt{d}(s', r')$, whereas the dual weight for s' remains unchanged. Therefore, the new dual weight will satisfy

$$y'(s') + y'(r') + (t-1)\mathtt{d}(s', r') = t\mathtt{d}(s', r'),$$

or

$$y'(s') + y'(r') = \mathtt{d}(s', r')$$

satisfying feasibility condition (2) with respect to matching M'.

For every other edge $(s', r') \in (S \times R) \setminus M'$, if r' is not on P, then the dual weights of s' and r' do not change and therefore the edge continues to be feasible. On the other hand, if r' is on the path P, suppose s'' is the match of r' in M'. the dual update will be as follows: $y'(r') \leftarrow y(r') - (t-1)\mathtt{d}(s'', r')$. Therefore,

$$y'(s') + y'(r') = y(s') + y(r') - (t-1)\mathtt{d}(s'', r') \leq \mathtt{d}(s', r').$$

The last inequality follows from the fact that $t \geq 1$, $\mathtt{d}(s'', r') \geq 0$, and the dual weights before augmentation satisfied (1). ◀

Proof of (I2). Initially, for every vertex $s' \in S$, its dual weight $y(s') = 0$. At the end of any iteration, we claim that the dual weight of every vertex in S_F remains 0. Recollect that our algorithm selects the free vertex $v \in S_F$ with the smallest d_v value. Therefore, for every other free vertex $v' \in S_F \setminus \{v\}$, $d_{v'} \geq d_v$ and therefore from (a), the dual weight of v' is not updated and remains 0. After augmentation, only the dual weights of points on P get updated. Since every vertex of P is matched after augmentation, there is no vertex of S_F on P. Therefore, the dual weight of every vertex of S_F remains 0.

For every vertex $v \in S$, initially $y(v) = 0$.In the update procedure for dual weights, if the vertex v belongs to S, its dual weight only reduces (see condition (c) for updating dual weight). The update procedure after augmentation, on the other hand, does not change the dual weight of any vertex in S. Therefore, the dual weight of v at any time during the algorithm is $y(v) \leq 0$. ◀

Efficiency. To process a new request, the algorithm executes Dijkstra's algorithm in $O(n^2)$ time and updates the matching and the dual weights by simply processing every node individually in $O(n)$ time.

▶ **Theorem 6.** *When a new request arrives, our algorithm processes and assigns an unmatched server to this request in $O(n^2)$ time.*

For $t = 1$, our algorithm is identical to the algorithm of Khuller *et al.* [10]. For $t > 1$, we are able to relate the cost of the online matching produced by our algorithm to the t-net-cost of the paths produced by the algorithm (Lemma 7(ii)). This relation is crucial to achieve desired performance bounds of our algorithm.

3 Performance of the Algorithm

To simplify the analysis of the algorithm, we introduce a few notations. We index the requests in the order of their arrival, i.e., let r_i be the ith request to arrive. To process request r_i, let P_i be the augmenting path computed by our algorithm. Let s_i be the free server at the other end of the augmenting path P_i. Let M_i^* be the offline matching after the ith request has been processed. Note that M_0^* is an empty matching and $M_n^* = M^*$ is the final matching after all the n requests have been processed. The online matching M_i is the online matching after i requests have been processed. M_i consists of edges $\bigcup_{j=1}^{i}(s_j, r_j)$.

For any path P, let $\ell(P) = \sum_{(s,r)\in P} \mathtt{d}(s,r)$ be its *length*. Next, we prove a useful relation between the t-net-cost of augmenting paths produced by our algorithm and the cost of the online matching. We utilize the metric property of costs to establish this relation.

▶ **Lemma 7.** *Let $t \geq 1$. Let $P_1, \dots, P_n$ be the augmenting paths computed by our algorithm in that order. Then, the t-net-cost of these paths relate to the cost of the online matching as follows:*

(i) $\phi_t(P_i) \leq t\mathtt{d}(s_i, r_i) \leq t\ell(P_i)$.

(ii) $\sum_{i=1}^{n} \phi_t(P_i) \geq ((t-1)/2)w(M) + ((t+1)/2)w(M^*)$.

Proof of (i). From triangle inequality, the length $\ell(P_i)$ of path P_i is at most the distance $\mathtt{d}(s_i, r_i)$ between its end-points. Therefore,

$$\mathtt{d}(s_i, r_i) \leq \ell(P_i). \tag{6}$$

With respect to matching M_{i-1}^*, the edge (s_i, r_i) is an augmenting path of length 1. The t-net-cost of this path is $t\mathtt{d}(a_i, b_i)$. Since, P_i is the minimum net-cost path with respect to M_{i-1}^*, $\phi_t(P_i) \leq t\mathtt{d}(a_i, b_i)$. This, combined with (6) implies (i). ◀

Proof of (ii). Since the matchings M_i^* and M_{i-1}^* differ only in the edges of the augmenting path P_i, we have

$$\begin{aligned} w(M_i^*) - w(M_{i-1}^*) &= \sum_{(s,r)\in P_i\setminus M_{i-1}^*} \mathtt{d}(s,r) - \sum_{(s,r)\in P_i\cap M_{i-1}^*} \mathtt{d}(s,r) \qquad (7)\\ &= \phi_t(P_i) - \left((t-1)\sum_{(s,r)\in P_i\setminus M_{i-1}^*} \mathtt{d}(s,r)\right)\\ &= \phi_t(P_i) - \left(\frac{t-1}{2}\sum_{(s,r)\in P_i\setminus M_{i-1}^*} \mathtt{d}(s,r) + \frac{t-1}{2}\sum_{(s,r)\in P_i\setminus M_{i-1}^*} \mathtt{d}(s,r)\right) \end{aligned}$$

The second equality follows from the definition of $\phi_t(\cdot)$.

We add and subtract $(\frac{t-1}{2})\sum_{(s,r)\in P_i\cap M^*_{i-1}} \mathtt{d}(s,r)$ to the RHS and get the following

$$\begin{aligned}
w(M^*_i) - w(M^*_{i-1}) &= \phi_t(P_i) - \frac{t-1}{2}\left(\sum_{(s,r)\in P_i\setminus M^*_{i-1}} \mathtt{d}(s,r) + \sum_{(s,r)\in P_i\cap M^*_{i-1}} \mathtt{d}(s,r)\right)\\
&\quad -\frac{t-1}{2}\left(\sum_{(s,r)\in P_i\setminus M^*_{i-1}} \mathtt{d}(s,r) - \sum_{(s,r)\in P_i\cap M^*_{i-1}} \mathtt{d}(s,r)\right)\\
&= \phi_t(P_i) - \frac{t-1}{2}(\sum_{(s,r)\in P_i} \mathtt{d}(s,r)) - \frac{t-1}{2}(w(M^*_i) - w(M^*_{i-1}))
\end{aligned}$$

The last equality follows from (7). Rearranging terms and setting $\sum_{(s,r)\in P_i}\mathtt{d}(s,r) = \ell(P_i)$, we get,

$$\begin{aligned}
\frac{t+1}{2}(w(M^*_i) - w(M^*_{i-1})) &= \phi_t(P_i) - \frac{t-1}{2}\ell(P_i)\\
\frac{t+1}{2}\sum_{i=1}^{n}(w(M^*_i) - w(M^*_{i-1})) &= \sum_{i=1}^{n}\phi_t(P_i) - \frac{t-1}{2}\sum_{i=1}^{n}\ell(P_i)\\
\frac{t+1}{2}w(M^*) &\le \sum_{i=1}^{n}\phi_t(P_i) - \frac{t-1}{2}w(M)
\end{aligned}$$

In the second to last equation, the summation on the LHS telescopes canceling all terms except $w(M^*_n) - w(M^*_0)$. Since $M^*_n = M^*$ and M^*_0 is an empty matching, we get $w(M^*_n) - w(M^*_0) = w(M^*)$. From triangle inequality, we know that $\ell(P_i) \ge \mathtt{d}(s_i, r_i)$. From this, we immediately get the last inequality. Rearranging the terms, we immediately get (ii). ◀

Next, equipped with the properties from Lemma 7, we will analyze the performance of our algorithm in the adversarial and the random arrival models.

To analyze the performance in the online models, let M^i_{OPT} be the minimum-cost matching of the first i requests to the set of servers. Also, we will denote M^n_{OPT} as M_{OPT}. It is easy to see that $w(M^i_{\text{OPT}}) \le w(M_{\text{OPT}})$. This is because M_{OPT} contains a matching of the first i request to servers S where as M^i_{OPT} is the smallest possible such matching. The cost of M^i_{OPT}, therefore, should be less than the cost of M_{OPT}.

$$w(M^i_{\text{OPT}}) \le w(M_{\text{OPT}}).$$

Performance in Adversarial Model. We show that the performance of our algorithm in the adversarial model is optimal.

▶ **Lemma 8.** *The competitive ratio of our algorithm in the adversarial model is* $2n - 1 + o(1)$.

Proof. Consider the graph $\tilde{G}$ with vertex set $S \cup R$ and the edges of the symmetric difference of M_{OPT} and M^*_{i-1}, i.e., $\tilde{G}(S\cup R, M_{\text{OPT}} \oplus M^*_{i-1})$. Since M_{OPT} is a perfect matching, this graph $\tilde{G}$ contains $n-i+1$ vertex-disjoint augmenting paths with respect to M^*_{i-1}. There is one augmenting path for each of the $n-i+1$ requests that have not yet arrived. Let $\{r'_1, r'_2, \ldots, r'_{n-i+1}\}$ be the requests that have not yet arrived and let the $n-i+1$ augmenting paths be $\{P'_1, P'_2, \ldots, P'_{n-i+1}\}$, where P'_j is an augmenting path that has r'_j as one of its end-vertex.

In particular, there is an augmenting path in $\tilde{G}$ with the i^{th} request r_i as one of its end vertex. Let P be this augmenting path.

$$\phi_t(P) = t \sum_{(s,r)\in M_{\text{OPT}}\cap P} \mathtt{d}(s,r) - \sum_{(s,r)\in M^*_{i-1}\cap P} \mathtt{d}(s,r) \le t \sum_{(s,r)\in M_{\text{OPT}}\cap P} \mathtt{d}(s,r) \le tw(M_{\text{OPT}})$$

While processing the ith request, our algorithm produces the minimum t-net-cost augmenting path. Therefore, the augmenting path P_i generated by our algorithm will have

$$\phi_t(P_i) \le \phi_t(P) \le tw(M_{\text{OPT}}).$$

If we sum over all the n augmenting paths generated by our algorithm, we get

$$\sum_{i=1}^{n} \phi_t(P_i) \le ntw(M_{\text{OPT}}).$$

Using property (ii) in Lemma 5 in conjunction with the previous inequality, we get the following

$$\begin{aligned}
ntw(M_{\text{OPT}}) &\ge ((t-1)/2)w(M) + ((t+1)/2))w(M^*) \\
2ntw(M_{\text{OPT}}) - (t+1)w(M^*) &\ge (t-1)w(M) \\
2ntw(M_{\text{OPT}}) - (t+1)w(M_{\text{OPT}}) &\ge (t-1)w(M) \\
w(M) &\le w(M_{\text{OPT}})(2nt-(t+1))/(t-1) \\
w(M)/w(M_{\text{OPT}}) &\le (2+2/(t-1))n - (1+2/(t-1))
\end{aligned}$$

The last inequality upper bounds the competitive ratio of the algorithm. If we set $t = n^2+1$, the upper bound can be simplified to $2n-1+1/n$ which is $2n-1+o(1)$. ◀

As shown in [8], no deterministic algorithm can achieve a competitive ratio better than $2n-1$. Therefore, our algorithm is optimal.

Performance in Random Arrival Model. In the random arrival model, we show that the performance ratio of our algorithm is $2\mathrm{H}_n - 1 + o(1)$.

▶ **Lemma 9.** *In the random arrival model, the competitive ratio of our algorithm is $2H_n - 1 + o(1)$.*

Proof. Consider the graph $\tilde{G}$ with vertex set $S \cup R$ and the edges of the symmetric difference of M_{OPT} and M^*_{i-1}, i.e., $\tilde{G}(S\cup R, M_{\text{OPT}} \oplus M^*_{i-1})$. Since M_{OPT} is a perfect matching, this graph contains $n-i+1$ vertex-disjoint augmenting paths with respect to M^*_{i-1}. There is one augmenting path for each of the $n-i+1$ requests that have not yet arrived. Let $\{r'_1, r'_2, \ldots, r'_{n-i+1}\}$ be the requests that have not yet arrived and let the $n-i+1$ augmenting paths in $\tilde{G}$ be $\{P'_1, P'_2, \ldots, P'_{n-i+1}\}$, where P'_j is an augmenting path that has r'_j as one of its end vertex.

$$\begin{aligned}
\sum_{j=1}^{n-i+1} \phi_t(P'_j) &= \sum_{j=1}^{n-i+1} \left(t \sum_{(s,r)\in P'_j \setminus M^*_{i-1}} \mathtt{d}(s,r) - \sum_{(s,r)\in P'_j\cap M^*_{i-1}} \mathtt{d}(s,r) \right) \\
&= \sum_{j=1}^{n-i+1} \left(t \sum_{(s,r)\in P'_j \cap M_{\text{OPT}}} \mathtt{d}(s,r) - \sum_{(s,r)\in P'_j\cap M^*_{i-1}} \mathtt{d}(s,r) \right) \\
&\le \sum_{j=1}^{n-i+1} \left(t \sum_{(s,r)\in P'_j \cap M_{\text{OPT}}} \mathtt{d}(s,r) \right) \le tw(M_{\text{OPT}}).
\end{aligned}$$

The second equation follows from the fact that these paths are formed by the symmetric difference of M^*_{i-1} and M_{OPT} and therefore $P'_j \setminus M^*_{i-1} = P'_j \cap M_{\text{OPT}}$. The last inequality follows from the fact that all the augmenting paths are vertex disjoint.

Let j be such that the i^{th} request r_i is r'_j. The algorithm computes the minimum t-net-cost augmenting path P_i from r_i with respect to the matching M^*_{i-1}. Therefore, the t-net-cost of P_i should be less than the t-net-cost of P'_j (note that r_i is one of the end-vertex of P'_j).

$$\phi_t(P_i) \leq \phi_t(P'_j).$$

In the random arrival model, the input request sequence is a random permutation. Therefore, the i^{th} request can be any one of the remaining $n-i+1$ requests with the same probability and we have,

$$\mathbb{E}[\phi_t(P_i)] \leq \frac{1}{n-i+1}\sum_{j=1}^{n-i+1}\phi_t(P'_j) \leq \frac{1}{n-i+1}tw(M_{\text{OPT}}).$$

From linearity of expectation,

$$\mathbb{E}[\sum_{i=1}^{n}\phi_t(P_i)] \quad \leq \quad \sum_{i=1}^{n}\frac{1}{n-i+1}tw(M_{\text{OPT}}) = t\mathrm{H}_n w(M_{\text{OPT}}).$$

From Lemma 7 (ii) and the obvious fact that $w(M^*) \geq w(M_{\text{OPT}})$ we have,

$$\begin{aligned}
\mathbb{E}[\sum_{i=1}^{n}\phi_t(P_i)] &\geq \frac{t-1}{2}\mathbb{E}[w(M)] + \frac{t+1}{2}\mathbb{E}[w(M^*)] \\
t\mathrm{H}_n w(M_{\text{OPT}}) &\geq \frac{t-1}{2}\mathbb{E}[w(M)] + \frac{t+1}{2}w(M_{\text{OPT}}) \\
(t-1)\mathbb{E}[w(M)] &\leq 2t\mathrm{H}_n w(M_{\text{OPT}}) - (t+1)w(M_{\text{OPT}}) \\
\frac{\mathbb{E}[w(M)]}{w(M_{\text{OPT}})} &\leq \frac{2t\mathrm{H}_n - (t+1)}{t-1} \\
\frac{\mathbb{E}[w(M)]}{w(M_{\text{OPT}})} &\leq (2+\frac{2}{t-1})\mathrm{H}_n - (1+\frac{2}{t-1})
\end{aligned}$$

By setting $t = nH_n + 1$, we can bound this competitive ratio by $2\mathrm{H}_n - 1 + o(1)$. ◀

▶ **Theorem 10.** *There is an algorithm for the online minimum metric bipartite matching problem that has a competitive ratio of $2n-1+o(1)$ in the adversarial model. This algorithm also has a competitive ratio of $2H_n - 1 + o(1)$ in the random arrival model.*

The performance of our algorithm is optimal in the random arrival model. A lower bound construction for the oblivious adversary model is described in [15]. We adapt this construction in the random arrival model. Obtaining a tight lower bound of $2\mathrm{H}_n - 1 - o(1)$ requires some technical calculations which we present next.

Lower Bound in the Random Arrival Model. Consider a undirected weighted star graph with a vertex v connected by an edge to every other vertex $v_1, \ldots v_n$ of this graph. Note that this graph has $n+1$ vertices and n edges. The weight of each of these n edges is 1. Consider the shortest path metric on this graph. For any pair (v, v_i), the shortest path distance is 1. Every other pair, (v_i, v_j) will have a shortest path distance of 2; this is because the only path between them goes via v and has cost 2. Given this graph, we place our servers at nodes $S = \{v_1, \ldots, v_n\}$. The adversary chooses request locations at nodes $R = \{v, v_1, \ldots v_n\} \setminus \{v_t\}$, where t is chosen uniformly at random from integers between 1 and n. Let σ be a random permutation of requests in R.

First, note that the minimum-cost matching of S and R is of cost 1 – the request at location v is matched to server at location v_t and every other server at location v_i is matched to the corresponding request at location v_i.

Consider any online algorithm for this input instance. Let us fix v to be the jth request in σ. All requests that arrived before v will be matched with zero cost to servers at the same location. It is easy to see that if the algorithm pursues any other matching scheme, it will only have a larger final cost. The cost of matching v is 1 since every server is at a distance 1 from v. Therefore, after processing the jth request, the total cost of the matching is 1.

Consider j'th request for any $j' > j$. Request $r_{j'}$ has to be in one of the remaining $n - j' + 1$ locations from the set $\{v_1, \dots, v_n\}$ that have not yet seen a request. Since t is chosen uniformly at random, the next request $r_{j'}$ can be any one of these $n - j' + 1$ locations with the same probability. Next, observe that, the servers in exactly one of these $n - j' + 1$ locations is already matched. If the next request is at this location, it will incur a cost of 2. This can happen with a probability of $1/(n - j' + 1)$. Therefore, the expected cost of serving the j'th request is at least $2/(n - j' + 1)$. Given that v was the jth request, the expected cost incurred will be at least

$$1 + \sum_{j'=j+1}^{n} (2/(n - j' + 1)) \quad \geq \quad 1 + 2\mathrm{H}_{n-j} = 1 + 2\ln(n - j) + 2\epsilon_{n-j} + 2\gamma, \tag{8}$$

where γ is the Euler–Mascheroni constant and $\epsilon_k \approx 1/2k$. Since σ is a random permutation, j can be any particular index between 1 and n with probability $1/n$. When $j = n$, the cost incurred by the algorithm is exactly 1. Equation 8 is meaningful only for $1 \leq j \leq n - 1$. Therefore, the expected cost incurred by any algorithm will be at least

$$\begin{aligned}
&\frac{1}{n}(1 + \sum_{j=1}^{n-1}(2\ln(n - j) + 2\epsilon_{n-j} + 2\gamma + 1)) \\
&= \frac{1}{n}(1 + 2\ln((n - 1)!) + n - 1 + 2\sum_{j=1}^{n} \epsilon_{n-j} + 2(n - 1)\gamma) \\
&= \frac{1}{n}(n + 2\ln((n - 1)!) + 2\sum_{j=1}^{n-1} \epsilon_{n-j} + 2(n - 1)\gamma) \\
&= 1 + \frac{2}{n}((n - 1)\ln(n - 1) - (n - 1) + (n - 1)\gamma + O(\log n)) \\
&\geq 1 + 2\ln n - 2 + \gamma + \epsilon_n - o(1) \\
&\geq 2\mathrm{H}_n - 2 + 1 - o(1) \\
&\geq 2\mathrm{H}_n - 1 - o(1)
\end{aligned}$$

The third equality follows from Sterling's approximation which gives us $\ln((n - 1)!) = (n - 1)\ln(n - 1) - (n - 1) + O(\log n)$. Also, $\sum_{j=1}^{n-1} \epsilon_{n-j} = O(\log n)$.

Therefore, in the random arrival model, any online matching generated by the algorithm will have an expected cost of $2\mathrm{H}_n - 1 - o(1)$ for this input.

▶ **Theorem 11.** *In the random arrival model, any online algorithm for the minimum metric bipartite matching problem will have a competitive ratio of at least $2H_n - 1 - o(1)$, where H_n is the nth Harmonic number.*

4 Conclusion

In this paper, we design a robust deterministic online algorithm for the minimum metric bipartite matching problem. Our algorithm achieves an optimal competitive ratio in both the adversarial and the random arrival models. We need such robust solutions for practically motivated real-time matching problems. We conclude with a few open questions:

(a) Can we extend our approach to the k-server problem and achieve better quality solutions?

(b) Can we improve the performance of our algorithm in special metrics such as the line metric or the Euclidean metric in $2d$?

(c) Can we extend our algorithm to the oblivious model and obtain a $O(\log n)$-competitive algorithm?

References

1 Pankaj K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31–June 03, 2014*, pages 555–564, 2014.

2 Pankaj K. Agarwal and Kasturi R. Varadarajan. A near-linear constant-factor approximation for euclidean bipartite matching? In *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*, pages 247–252, 2004.

3 Antonios Antoniadis, Neal Barcelo, Michael Nugent, Kirk Pruhs, and Michele Scquizzato. A o(n)-competitive deterministic algorithm for online matching on a line. In *Approximation and Online Algorithms – 12th International Workshop, WAOA 2014, Wrocław, Poland, September 11-12, 2014*, pages 11–22, 2014.

4 N. Bansal, N. Buchbinder, A Madry, and J. Naor. A polylogarithmic-competitive algorithm for the k-server problem. In *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 267–276, Oct 2011.

5 Nikhil Bansal, Niv Buchbinder, Anupam Gupta, and Joseph Naor. An $o(\log^2 k)$-competitive algorithm for metric bipartite matching. In *Algorithms – ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 522–533, 2007.

6 A. Gupta and K. Lewi. The online metric matching problem for doubling metrics. In *Automata, Languages, and Programming*, volume 7391 of *LNCS*, pages 424–435. Springer, 2012.

7 Piotr Indyk. A near linear time constant factor approximation for euclidean bichromatic matching (cost). In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 39–42, 2007.

8 Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *J. Algorithms*, 14(3):478–488, 1993.

9 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*, STOC'11, pages 587–596, New York, NY, USA, 2011. ACM.

10 Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theor. Comput. Sci.*, 127(2):255–267, 1994.

11 Elias Koutsoupias and Akash Nanavati. The online matching problem on a line. In Roberto Solis-Oba and Klaus Jansen, editors, *Approximation and Online Algorithms: First International Workshop, WAOA 2003, Budapest, Hungary, September 16-18, 2003. Revised Papers*, pages 179–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

12 Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, September 1995.

13 M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, STOC'11, pages 597–606, 2011.

14 Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230, May 1990.

15 A. Meyerson, A. Nanavati, and L. Poplawski. Randomized online algorithms for minimum metric bipartite matching. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 954–959, 2006.

16 R. Sharathkumar and Pankaj K. Agarwal. Algorithms for the transportation problem in geometric settings. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 306–317, 2012.

17 R. Sharathkumar and Pankaj K. Agarwal. A near-linear time ϵ-approximation algorithm for geometric bipartite matching. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC'12, pages 385–394. ACM, 2012. doi:10.1145/2213977.2214014.

Search-to-Decision Reductions for Lattice Problems with Approximation Factors (Slightly) Greater Than One[*]

Noah Stephens-Davidowitz

Courant Institute of Mathematical Sciences, New York University, NY, USA
noahsd@gmail.com

Abstract

We show the first dimension-preserving search-to-decision reductions for approximate SVP and CVP. In particular, for any $\gamma \leq 1 + O(\log n/n)$, we obtain an efficient dimension-preserving reduction from $\gamma^{O(n/\log n)}$-SVP to γ-GapSVP and an efficient dimension-preserving reduction from $\gamma^{O(n)}$-CVP to γ-GapCVP. These results generalize the known equivalences of the search and decision versions of these problems in the exact case when $\gamma = 1$. For SVP, we actually obtain something slightly stronger than a search-to-decision reduction – we reduce $\gamma^{O(n/\log n)}$-SVP to γ-unique SVP, a potentially easier problem than γ-GapSVP.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical Problems and Computations

Keywords and phrases Lattices, SVP, CVP

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.19

1 Introduction

A lattice $\mathcal{L} = \{\sum a_i \mathbf{b}_i : a_i \in \mathbb{Z}\} \subset \mathbb{R}^n$ is the set of all integer linear combinations of linearly independent basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$.

The two most important computational problems on lattices are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). For any approximation factor $\gamma = \gamma(n) \geq 1$, γ-SVP is the search problem that takes as input a lattice and asks us to find a non-zero vector in this lattice whose length is within a factor of γ of the minimal possible value. γ-CVP is the search problem that takes as input both a lattice and a target vector $\mathbf{t} \in \mathbb{R}^n$ and asks us to find a vector in $\mathcal{L}$ whose distance to $\mathbf{t}$ is within a factor of γ of the minimal distance. The natural decisional variants of these problems are called GapSVP and GapCVP respectively. Specifically, γ-GapSVP asks us to approximate the length of the shortest non-zero vector of a lattice up to a factor of γ, and γ-GapCVP asks us to approximate the distance from $\mathbf{t}$ to the lattice up to a factor of γ.

All four of these problems are interesting for a wide range of approximation factors γ. Indeed, algorithms for these problems have found a remarkable number of applications in computer science (e.g., [26, 27, 23, 36, 21, 35, 15]). And, over the past twenty years, many strong cryptographic primitives have been constructed with their security based on the (worst-case) hardness of γ-GapSVP with approximation factors $\gamma = \text{poly}(n)$ that are polynomial in the ambient dimension (e.g., [4, 34, 18, 17, 37, 40, 30, 11, 12]).

[*] Supported by the National Science Foundation (NSF) under Grant No. CCF-1320188. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016).
Editors: Klaus Jansen, Claire Matthieu, José D. P. Rolim, and Chris Umans; Article No. 19; pp. 19:1–19:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Due to their importance, there has been much work towards understanding the relationship between these problems (and their many close relatives). Since the fastest known algorithms for these problems run in time that is exponential in the dimension n, even with $\gamma = \mathrm{poly}(n)$, *dimension-preserving* reductions between lattice problems are of particular importance [23, 19, 32, 29, 41, 42]. Perhaps the best-known such reduction is the efficient dimension-preserving reduction from γ-SVP to γ-CVP (and from γ-GapSVP to γ-GapCVP) due to Goldreich, Micciancio, Safra, and Seifert [19]. This proves that the time complexity of γ-SVP, as a function of the dimension n, cannot be more than a polynomial factor higher than the time complexity of γ-CVP. We stress that we could *not* reach this conclusion if the reduction increased the dimension significantly, which is why dimension-preserving reductions interest us.

As a much simpler example, we note that there is a trivial dimension-preserving reduction from γ-GapSVP to γ-SVP that works by just finding a short vector in the input lattice and outputting its length. There is of course a similar reduction for CVP as well. More interestingly, there are relatively simple dimension-preserving *search-to-decision* reductions in the special case when $\gamma = 1$ – i.e., finding *exact* shortest vectors is no harder than computing the *exact* lengths of shortest vectors, and finding exact closest vectors to targets is no harder than computing the exact distances between targets and lattices. (See, e.g., [23] or [33], or simply consider the reductions in the sequel with $\gamma = 1$.) However, prior to this work, there were no known search-to-decision reductions for either SVP or CVP for any approximation factor $\gamma > 1$.

This state of affairs was quite frustrating because, with very few exceptions, our best algorithms for the decision problems work by just solving the corresponding search problem. In other words, we don't really know how to "recognize" that a lattice has a short non-zero vector (or a vector close to some target) without just finding such a vector.[1] If there are better techniques, then we would be thrilled to find them! But, if this extremely natural approach is actually optimal, then it would be nice to prove it by formally reducing the search problems to their decision variants. (Of course, it is conceivable that the search and decision problems have the same complexity, even if no search-to-decision reduction exists. One might reasonably argue that this is even the most likely scenario. But, we can at least hope that Nature would not be so unprincipled.)

The ideal positive result would be an efficient dimension-preserving reduction from γ-SVP to γ-GapSVP for all $\gamma \geq 1$, and likewise for CVP. But, this seems completely out of reach at the moment (perhaps because no such reductions exist). So, as a more approachable goal, we can try to find non-trivial reductions that lose in the approximation factor. Indeed, as we mentioned above, we know that search and decision problems are equivalent in the exact case. Can it truly be the case that equivalence holds when $\gamma = 1$, but *nothing* non-trivial holds for any $\gamma > 1$ – even, say, a reduction from n^{100}-CVP to $(1 + 2^{-n})$-GapCVP?!

1.1 Our results

We make some progress towards resolving these issues by presenting dimension-preserving search-to-decision reductions for both approximate SVP and approximate CVP. Our reduc-

[1] The author knows of three rather specific exceptions. There is an efficient algorithm for $\sqrt{n/\log n}$-GapCVP *with preprocessing* [3], while the best efficient algorithm for search CVP with preprocessing only achieves factor of $\gamma = n/\sqrt{\log n}$ [16]. There is a $2^{n/2+o(n)}$-time algorithm for 2-GapSVP for which no analogous search algorithm is known [1]. And, in the special case of ideal lattices in the ring of integers of a number field, γ-GapSVP is trivial for some values of γ for which γ-SVP appears to be hard. (See, e.g., [38].)

tions generalize the known equivalences in the exact case. But, they lose quite a bit in the approximation factor, and their running times depend on the decision approximation factor. They are therefore primarily interesting when the decision approximation factor is very close to one, as we explain below.

▶ **Theorem 1** (SVP reduction)**.** *For any $\gamma = \gamma(n) \geq 1$ and $a = a(n) \geq \log(n+1)$, there is a dimension-preserving (randomized) reduction from $\gamma^{n/a}$-SVP to γ-GapSVP that runs in time $2^{O(a)} \cdot \gamma^{O(n)}$.*

Theorem 1 is primarily interesting for any $\gamma \leq 1 + O(\log n/n)$ and $a = \Theta(\log n)$. For such parameters, the running time is $\mathrm{poly}(n)$ and the search approximation factor is $\gamma^{O(n/\log n)} \leq O(1)$. However, we note that the theorem is non-trivial whenever we have $1 < \gamma \leq 1 + \varepsilon$ and $a \leq \varepsilon n$, where $\varepsilon > 0$ is some small universal constant.[2]

We actually reduce $\gamma^{n/a}$-SVP to γ-unique SVP, which is a potentially easier problem than γ-GapSVP. (See Definition 16 for the formal definition of γ-unique SVP, and Theorem 24 for the reduction.) The reduction described above then follows from this result together with Lyubashevsky and Micciancio's reduction from γ-unique SVP to γ-GapSVP [29]. We obtain a few additional corollaries as well. E.g., this shows a dimension-preserving reduction from $\sqrt{n}$-CVP to γ-unique SVP (and thus to γ-GapSVP as well) that runs in time $\mathrm{poly}(n) \cdot \gamma^{O(n)}$. This also gives an alternative and arguably more natural proof of Aggarwal and Dubey's result that γ-unique SVP is NP-hard (under randomized reductions) for $\gamma \leq 1 + 1/n^{\varepsilon}$ for any constant $\varepsilon > 0$ [2].

With some more work, we are also able to use our SVP reduction to derive the following search-to-decision reduction for CVP.

▶ **Theorem 2** (CVP reduction)**.** *For any $\gamma = \gamma(n) \geq 1$ and $\ell = \ell(n) \geq 1$, there is a dimension-preserving (randomized) reduction from $\gamma^{n/\ell}$-CVP to γ-GapCVP that runs in time $n^{O(\ell)} \cdot \gamma^{O(n)}$.*

This result is primarily interesting when ℓ is any constant and $\gamma \leq 1 + O(\log n/n)$, in which case the reduction runs in polynomial time and the search approximation factor is $\gamma^{O(n)} \leq \mathrm{poly}(n)$. But, it is still non-trivial for $1 < \gamma \leq 1 + \varepsilon$ and $\ell \leq \varepsilon n/\log n$, where $\varepsilon > 0$ is some universal constant.

We actually show a (deterministic) $n^{O(\ell)}$-time reduction from $\gamma^{n/\ell}$-CVP to γ-GapCVP that works in the presence of a $\mathrm{poly}(n)$-SVP oracle. (See Theorem 28.) The above result then follows from instantiating this oracle via our SVP reduction.

Finally, we show deterministic reductions that achieve much worse parameters.

▶ **Theorem 3** (Deterministic SVP reduction)**.** *For any $\gamma = \gamma(n) \geq 1$ and $p = p(n) \geq 2$, there is a deterministic dimension-preserving reduction from γ'-SVP to γ-GapSVP that runs in time $\mathrm{poly}(n) \cdot p$, where $\gamma' := \gamma^{O(n^2/\log p)}$.*

▶ **Theorem 4** (Deterministic CVP reduction)**.** *For any $\gamma = \gamma(n) \geq 1$ and $p = p(n) \geq 2$, there is a deterministic dimension-preserving reduction from γ'-CVP to γ-GapCVP that runs in time $\mathrm{poly}(n) \cdot p$, where $\gamma' := \gamma^{O(n^2/\log p)}$.*

It is easy to see that our randomized reductions always give a better trade-off between the approximation factor and running time for non-trivial parameters. So, these new reductions

[2] In particular, we can choose ε so that, with $a = \varepsilon n$ and $\gamma \leq 1 + \varepsilon$, we get a reduction from $\gamma^{1/\varepsilon}$-SVP to γ-GapSVP that runs in time $O(2^n)$. For larger values of a or γ, the reduction is subsumed by the known $2^{n+o(n)}$-time algorithm for SVP [1].

are primarily interesting because they are deterministic and because they demonstrate additional potential approaches for future work in this area.

We note that all of our reductions are Cook reductions. (They make many oracle calls, sometimes adaptively.)

1.2 Techniques

1.2.1 SVP

Our main SVP reduction works by finding a sublattice of the input lattice that has one relatively short vector but a significantly longer "second-shortest vector." (I.e., we wish to find a sublattice that satisfies the promise of γ-unique SVP. See Definition 16.) To accomplish this, we use lattice sparsification, which was introduced by Khot [24] and refined in a series of works [14, 16, 42].

The idea of sparsification is to consider the "sparsified" sublattice

$$\mathcal{L}' := \{\mathbf{y} \in \mathcal{L} \,:\, \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{y}\rangle \equiv 0 \bmod p\} \ ,$$

where p is some prime and $\mathbf{z} \in \mathbb{Z}_p^n$ is chosen uniformly at random. We would like to say that each short vector in $\mathcal{L}$ will land in $\mathcal{L}'$ with probability roughly $1/p$, independently of all other short vectors. Of course, if $\mathbf{x}, \mathbf{y} \in \mathcal{L}$ and $\mathbf{x} = k\mathbf{y}$ for some $k \not\equiv 0 \bmod p$, then clearly $\mathbf{x}$ will land in $\mathcal{L}'$ if and only if $\mathbf{y}$ does. So, we cannot have anything close to independence in this case. [42] shows that this is essentially "the only bad case."

Specifically, a lattice vector $\mathbf{x} \in \mathcal{L}$ is *non-primitive* if $\mathbf{x} = k\mathbf{y}$ for some $k \geq 2$ and $\mathbf{y} \in \mathcal{L}$. Otherwise, $\mathbf{x}$ is *primitive.* [42] showed that sparsification behaves very nicely if we restrict our attention to primitive short lattice vectors. In particular, the distribution of short primitive vectors in the sparsified sublattice $\mathcal{L}'$ behaves similarly to the distribution over the short primitive vectors of $\mathcal{L}$ that selects each vector independently with probability $1/p$. (See Theorem 22 for the precise statement, which is taken directly from [42, Theorem 4.1].)

So, let $\xi(\mathcal{L}, r)$ be the number of primitive lattice points of length at most r. Suppose there exists some radius r such that $\xi(\mathcal{L}, \gamma r)$ is not much larger than $\xi(\mathcal{L}, r)$.[3] Then, if we take $p \approx \xi(\mathcal{L}, \gamma r)$, we can expect $\mathcal{L}'$ to contain a primitive vector of length at most r *but no other primitive vectors of length less than* γr with probability $\Theta(\xi(\mathcal{L}, r)/\xi(\mathcal{L}, \gamma r))$. In other words, with this probability, $\mathcal{L}'$ will be a valid instance of γ-unique SVP with $\lambda_1(\mathcal{L}') \leq r$, so that we can use an oracle for γ-unique SVP to find a non-zero lattice vector of length at most r.

The parameter a in the reduction determines how large of a ratio $\xi(\mathcal{L}, \gamma r)/\xi(\mathcal{L}, r)$ we are "willing to tolerate." In particular, a simple proof shows that, for $a \geq n \log \gamma$, there is always a radius $r \leq \gamma^{n/a} \cdot \lambda_1(\mathcal{L})$ such that this ratio is bounded by $2^{O(a)}$. We can therefore obtain a valid γ-unique SVP instance with $\lambda_1(\mathcal{L}') \leq r \leq \gamma^{n/a} \cdot \lambda_1(\mathcal{L})$ with probability at least $2^{-O(a)}$. So, our main reduction essentially works by sampling $\mathcal{L}'$ repeatedly, a total of $2^{O(a)}$ times, and calling its γ-unique SVP oracle on each sampled sublattice $\mathcal{L}'$.

1.2.2 CVP

Our search-to-decision reduction for CVP is a simple "guided" variant of Babai's celebrated nearest-hyperplane algorithm [7]. Babai's algorithm works by dividing the lattice into $(n-1)$-

[3] It might be helpful to think of the heuristic that $\xi(\mathcal{L}, \gamma r)/\xi(\mathcal{L}, r) \approx \gamma^n$. This holds in the limit as $r \to \infty$, and it holds for random lattices in expectation.

dimensional lattice hyperplanes and then searching inside the closest hyperplane to the target. However, it is possible that the closest lattice hyperplane does not actually contain very close lattice points. As a result, Babai's algorithm can lead to quite large approximation factors.

So, we instead use our GapCVP oracle to "test many nearby hyperplanes" to find one that is guaranteed to contain a γ-approximate closest lattice point. By repeating this n times over hyperplanes of progressively lower dimensions, we will find a γ^n-approximate closest vector to the target. To find a $\gamma^{n/\ell}$-approximate closest vector, we do the same thing with all nearby $(n - \ell)$-dimensional hyperplanes.

In order to make this algorithm efficient, we need to limit the number of hyperplanes that we must consider. This amounts to finding a short non-zero vector in the dual lattice. We can find such a vector by using our search-to-decision reduction for SVP (together with the known reduction from GapSVP to GapCVP [19]). Unfortunately, this costs us a factor of $\gamma^{O(n)}$ in the running time.

1.2.3 Deterministic reductions

Our alternative deterministic search-to-decision reductions for SVP and CVP are very similar to the reduction from unique SVP to GapSVP in [29]. They essentially work by finding the coordinates of a short (or close) lattice vector "bit by bit." I.e., in the CVP case, we first use our GapCVP oracle to compare the distance from the target to all lattice vectors whose last coordinate is even with its distance from all lattice vectors whose last coordinate is odd. If, say, the odd estimate is lower, then we restrict our attention to the lattice coset of all lattice vectors whose last coordinate is odd. We choose the remaining bits similarly, eventually obtaining the coordinates of a relatively close lattice vector. Our more general reductions follow from "working in base p instead of base 2."

1.3 Related work

Some efficient dimension-preserving search-to-decision reductions were known for other lattice problems prior to this work. For example, Regev showed such a reduction for Learning with Errors, an important average-case lattice problem with widespread applications in cryptography [40]. (Both the search and decision versions of LWE are average-case problems.) And, Liu, Lyubashevsky, and Micciancio implicitly use a search-to-decision reduction for Bounded Distance Decoding in their work [28]. Finally, Aggarwal and Dubey showed how to use some of the ideas from [29] to obtain a search-to-decision reduction for unique SVP [2]. While all of these works are quite interesting, they are concerned with promise problems, and not the two most important and natural lattice problems, SVP and CVP.

More generally, this work can be seen as part of the ongoing study of the relationships between lattice problems under dimension-preserving reductions. By now, this area has become quite fruitful (e.g., [23, 19, 32, 29, 42]). See [41] for a brief survey of well-known dimension-preserving reductions between various lattice problems.

Most prior work used sparsification to remove a relatively small number of "annoying" short vectors from a lattice without losing too many "good" short vectors (e.g., [24, 14, 16]). In our main SVP reduction, our goal is instead to remove "all but one" short vector. (Independent work of Bai, Wen, and Stehlé used sparsification in a similar way to reduce Bounded Distance Decoding to unique SVP [8].) To obtain our result, we rely heavily on the sparsification analysis of [42], which is tighter and more general than prior work.

Interestingly, Kumar and Sivakumar used a procedure that is very similar to sparsification in their study of unique SVP, published in 2001 [25]. Indeed, they repeatedly sparsify a lattice with $p = 2$ to obtain a sequence of sublattices such that at least one of these sublattices (1) contains a shortest non-zero vector of the original lattice; and (2) contains no other vectors of this length (up to sign, of course). However, the length of the "second-shortest vector" can be arbitrarily close to that of the shortest vector in their construction, even in a fixed dimension n. I.e., they use a restricted form of sparsification to effectively reduce 1-SVP to 1-unique SVP. Our main SVP reduction can be thought of as an updated version of their result. We use tools that were not available fifteen years ago to obtain a lower bound on the ratio between the shortest vector and the "second-shortest vector" that depends only on the dimension n.

To prove hardness of $(1 + 1/\mathrm{poly}(n))$-unique SVP, Aggarwal and Dubey used the result of Kumar and Sivakumar to show a reduction from SVP to γ-unique SVP that works for a restricted subset of lattices [2]. In particular, Aggarwal and Dubey chose a set of lattices such that, over these lattices (1) SVP is NP-hard (as proven by Khot [24]); and (2) this reduction yields $\gamma = 1 + 1/\mathrm{poly}(n)$. In contrast, we directly reduce 2-SVP to $(1 + 1/\mathrm{poly}(n))$-unique SVP over *all* lattices by using a much stronger (and unfortunately more complicated) form of Kumar and Sivakumar's reduction.

While the author knows of no other use of our specific variant of Babai, we feel that it is quite natural and not particularly novel. For example, a similar idea was used in a different context by Micciancio [32, Corollary 7]. Our primary contribution on this front is the observation that this method gives a non-trivial search-to-decision reduction when the decision approximation factor is very small, and when it is combined with our SVP reduction.

We rely heavily on Lyubashevsky and Micciancio's dimension-preserving reduction from γ-unique SVP to γ-GapSVP [29]. Their result is necessary to prove Theorem 1, and our deterministic "bit-by-bit" SVP reduction is very similar to Lyubashevsky and Micciancio's reduction. The main difference between our deterministic SVP reduction and that of Lyubashevsky and Micciancio is that [29] work only with lattices that satisfy the promise of γ-unique SVP. They show that this promise is enough to guarantee that the γ-GapSVP oracle essentially behaves as an *exact* GapSVP oracle. In contrast, our reduction works over general lattices, so we have to worry about accumulating error. (We also use a different method to "reduce the dimension of the lattice.")

1.4 Directions for future work

We view this paper as a first step towards a better understanding of the relationship between the search and decision variants of approximate SVP and CVP. In particular, we show that efficient, dimension-preserving search-to-decision reductions do in fact exist for approximation factors $\gamma > 1$. Prior to this work, one might have reasonably conjectured that such reductions do not exist for non-trivial parameters. But, our reductions lose quite a bit in the approximation factor, and the running times of our main reductions blow up quickly as the approximation factor increases. They are therefore primarily interesting for very small approximation factors $\gamma = 1 + o(1)$.

Results for such low values of γ have sometimes led to similar results for larger approximation factors. For example, hardness of γ-GapSVP was originally proven for $\gamma = 1 + 2^{-\mathrm{poly}(n)}$ [5], and then for $\gamma = 1 + 1/\mathrm{poly}(n)$ [13], before better inapproximability results were found [31, 24, 20]. We therefore ask whether better search-to-decision reductions exist, and in particular, whether non-trivial efficient dimension-preserving reductions exist for larger approximation factors.

More specifically, we note that our main reductions are only efficient when the decision approximation factor is $\gamma = 1 + O(\log n/n)$ because their running time is proportional to

$\gamma^{O(n)}$. This seems inherent to our technique in the case of SVP, and the CVP reduction suffers the same fate because it uses the SVP reduction as a subroutine. However, we see no reason why the running time should necessarily increase with the approximation factor, and this might simply be an artifact of our techniques. So, perhaps we can find reductions that do not have this problem. (One might try, for example, to eliminate the need for the SVP oracle in our CVP reduction.) Indeed, the reductions in Section 5 manage to avoid this pitfall, but they blow up the approximation factor much more and never actually outperform our main reductions.

In the other direction, we ask whether the search and decision versions of SVP and CVP can be separated in any way. I.e., can we show that, for some $\gamma > 1$, there is no efficient dimension-preserving reduction from γ-CVP to γ-GapCVP or no such reduction from γ-SVP to γ-GapSVP (under reasonable complexity-theoretic assumptions or even restrictions on the behavior of the reduction)? Can we find algorithms that solve the decision problems faster than our current search-based techniques allow (something more general than the rather specific examples mentioned in footnote 1)? Of course, any such result would be a major breakthrough.

2 Preliminaries

We write $\log x$ for the logarithm of x in base 2. We write $\|\mathbf{x}\|$ for the Euclidean norm of $\mathbf{x} \in \mathbb{R}^n$. We omit any mention of the bit length of the input throughout. In particular, all of our algorithms take as input vectors in $\mathbb{R}^n$ (with some reasonable representation) and run in time $f(n) \cdot \mathrm{poly}(m)$ for some f, where m is the maximal bit length of an input vector. We are primarily interested in the dependence on n, so we suppress the factor of $\mathrm{poly}(m)$.

2.1 Lattice basics

A rank d lattice $\mathcal{L} \subset \mathbb{R}^n$ is the set of all integer linear combinations of d linearly independent vectors $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$. $\mathbf{B}$ is called a basis of the lattice and is not unique. We write $\mathcal{L}(\mathbf{B})$ to signify the lattice generated by $\mathbf{B}$. By taking the ambient space to be $\mathrm{span}(\mathcal{L})$, we can implicitly assume that a lattice has full rank n, and we therefore will often implicitly assume that $d = n$.

The dual lattice is

$$\mathcal{L}^* := \{\mathbf{w} \in \mathrm{span}(\mathcal{L}) : \forall \mathbf{y} \in \mathcal{L}, \langle \mathbf{w}, \mathbf{y} \rangle \in \mathbb{Z}\} .$$

Similarly, the dual basis $\mathbf{B}^* := \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1} = (\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*)$ is the unique list of vectors in $\mathrm{span}(\mathcal{L})$ satisfying $\langle \mathbf{b}_i^*, \mathbf{b}_j \rangle = \delta_{i,j}$. $\mathcal{L}^*$ is itself a rank d lattice with basis $\mathbf{B}^*$.

We write $\lambda_1(\mathcal{L}) := \min_{\mathbf{x} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{x}\|$ for the length of the shortest non-zero vector in the lattice. Similarly, we write $\lambda_2(\mathcal{L}) := \min\{r > 0 \ : \ \dim(\mathrm{span}(\mathcal{L} \cap rB_2^n)) \geq 2\}$ for the length of the shortest lattice vector that is linearly independent from a lattice vector of length $\lambda_1(\mathcal{L})$. For any point $\mathbf{t} \in \mathbb{R}^n$, we write $\mathrm{dist}(\mathbf{t}, \mathcal{L}) := \min_{\mathbf{x} \in \mathcal{L}} \|\mathbf{x} - \mathbf{t}\|$ for the distance between $\mathbf{t}$ and $\mathcal{L}$. The covering radius $\mu(\mathcal{L}) := \max_{\mathbf{t} \in \mathrm{span}(\mathcal{L})} \mathrm{dist}(\mathbf{t}, \mathcal{L})$ is the maximal such distance achievable in the span of the lattice.

The following two bounds will be useful.[4]

[4] We note that tighter bounds exist for the number of lattice points in a ball of radius r [22, 39], but we use the bound of [10] because it is simpler. Using a tighter bound here would improve the hidden constants in the exponents of our running times.

▶ **Theorem 5** ([10, Theorem 2.1]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $r > 0$,*

$$|\{\mathbf{y} \in \mathcal{L} : \|\mathbf{y}\| \le r\lambda_1(\mathcal{L})\}| \le 2\lceil 2r \rceil^n - 1 .$$

▶ **Lemma 6** ([9, Theorem 2.2]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\lambda_1(\mathcal{L}^*) \cdot \mu(\mathcal{L}) \le n/2$.*

We derive a simple though rather specific corollary of Lemma 6 that we will use twice. The corollary says that a dual vector $\mathbf{w} \in \mathcal{L}^* \setminus \{\mathbf{0}\}$ that is relatively short, $\|\mathbf{w}\| \le \gamma \cdot \lambda_1(\mathcal{L}^*)$, can be used to partition $\mathcal{L}$ into $(n-1)$-dimensional lattice hyperplanes, such that the closest vector to any target $\mathbf{t}$ must lie in one of the $O(\gamma n)$ hyperplanes closest to $\mathbf{t}$.

▶ **Corollary 7.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and associated dual basis $(\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$, $\gamma \ge 1$, and any target $\mathbf{t} \in \mathbb{R}^n$, if $\|\mathbf{b}_1^*\| \le \gamma \cdot \lambda_1(\mathcal{L}^*)$, then any closest lattice vector to $\mathbf{t}$ must lie in a lattice hyperplane $\mathcal{L}' + i\mathbf{b}_1$, where $\mathcal{L}' := \mathcal{L}(\mathbf{b}_2, \ldots, \mathbf{b}_n)$ and i is an integer with $|i - \langle \mathbf{b}_1^*, \mathbf{t} \rangle| \le \gamma n/2$.*

Proof. Let $\mathbf{y} \in \mathcal{L}$ be a closest lattice vector to $\mathbf{t}$. It follows from the definition of a lattice that $\mathbf{y} \in \mathcal{L}' + i\mathbf{b}_1$ for some integer $i = \langle \mathbf{b}_1^*, \mathbf{y} \rangle$. We have

$$|i - \langle \mathbf{b}_1^*, \mathbf{t} \rangle| = |\langle \mathbf{b}_1^*, \mathbf{y} - \mathbf{t} \rangle| \le \|\mathbf{b}_1^*\| \|\mathbf{y} - \mathbf{t}\| \le \gamma \lambda_1(\mathcal{L}^*) \cdot \mu(\mathcal{L}) \le \gamma n/2 ,$$

where we have used Lemma 6. ◀

2.2 LLL-reduced bases

Given a basis, $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, we define its Gram-Schmidt orthogonalization $(\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_n)$ by

$$\widetilde{\mathbf{b}}_i = \pi_{\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}^\perp}(\mathbf{b}_i) ,$$

and the Gram-Schmidt coefficients $\mu_{i,j}$ by

$$\mu_{i,j} = \frac{\langle \mathbf{b}_i, \widetilde{\mathbf{b}}_j \rangle}{\|\widetilde{\mathbf{b}}_j\|^2} .$$

Here, π_A represents orthogonal projection onto the subspace A and $\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}^\perp$ denotes the subspace of vectors in $\mathbb{R}^n$ that are orthogonal to $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.

▶ **Definition 8.** A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is *LLL-reduced* if

1. for $1 \le j < i \le n$, $|\mu_{i,j}| \le 1/2$; and
2. for $2 \le i \le n$, $\|\widetilde{\mathbf{b}}_i\|^2 \ge (3/4 - \mu_{i,i-1}^2) \cdot \|\widetilde{\mathbf{b}}_{i-1}\|^2$.

▶ **Theorem 9** ([26]). *There exists an efficient algorithm that takes as input a (basis for) a lattice and outputs an LLL-reduced basis for the lattice.*

▶ **Lemma 10.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with LLL-reduced basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and $\mathbf{y} = \sum a_i \mathbf{b}_i \in \mathcal{L}$, we have*

$$|a_i| \le 2^{3n/2 - i} \cdot \frac{\|\mathbf{y}\|}{\lambda_1(\mathcal{L})} ,$$

for all i.

Proof. It follows immediately from the definition of an LLL-reduced basis that $\|\widetilde{\mathbf{b}}_i\| \geq \|\mathbf{b}_1\|/2^{i/2} \geq \lambda_1(\mathcal{L})/2^{n/2}$ for all i. For each i, we have

$$\|\mathbf{y}\| \geq \sum_{j=1}^{n} |a_j \mu_{j,i}| \cdot \|\widetilde{\mathbf{b}}_i\| = \Big(|a_i| - \sum_{j=i+1}^{n} |a_j \mu_{j,i}|\Big) \cdot \|\widetilde{\mathbf{b}}_i\| \geq \Big(|a_i| - \frac{1}{2} \sum_{j=i+1}^{n} |a_j|\Big) \cdot 2^{-n/2} \cdot \lambda_1(\mathcal{L}) \; .$$

In particular, $|a_n| \leq 2^{n/2} \cdot \|\mathbf{y}\|/\lambda_1(\mathcal{L})$. We assume for induction that $|a_j| \leq 2^{3n/2-j} \cdot \|\mathbf{y}\|/\lambda_1(\mathcal{L})$ for all j with $i < j \leq n$. Then, plugging in to the above, we have

$$\|\mathbf{y}\| \geq |a_i| \cdot 2^{-n/2} \cdot \lambda_1(\mathcal{L}) - \sum_{j=i+1}^{n} 2^{n-j-1} \|\mathbf{y}\| \geq |a_i| \cdot 2^{-n/2} \cdot \lambda_1(\mathcal{L}) - 2^{n-i-1} \cdot \|\mathbf{y}\| \; .$$

The result follows by rearranging. ◀

▶ **Lemma 11** ([7]). *If $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is an LLL-reduced basis, then $\mu(\mathcal{L}) \leq \sqrt{n}2^{n/2-1} \cdot \|\widetilde{\mathbf{b}}_n\|$.*

2.3 Lattice problems

We now list the computational problems that concern us. All of the below definitions are standard.

▶ **Definition 12.** For any parameter $\gamma = \gamma(n) \geq 1$, γ-SVP (the Shortest Vector Problem) is the search problem defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$. The goal is to output a lattice vector $\mathbf{x}$ with $0 < \|\mathbf{x}\| \leq \gamma\lambda_1(\mathcal{L})$.

▶ **Definition 13.** For any parameter $\gamma = \gamma(n) \geq 1$, γ-CVP (the Closest Vector Problem) is the search problem defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a target vector $\mathbf{t} \in \mathbb{R}^n$. The goal is to output a lattice vector $\mathbf{x}$ with $\|\mathbf{x} - \mathbf{t}\| \leq \gamma \operatorname{dist}(\mathbf{t}, \mathcal{L})$.

▶ **Definition 14.** For any parameter $\gamma = \gamma(n) \geq 1$, the decision problem γ-GapSVP is defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a number $d > 0$. The goal is to output yes if $\lambda_1(\mathcal{L}) < d$ and no if $\lambda_1(\mathcal{L}) \geq \gamma \cdot d$.

▶ **Definition 15.** For any parameter $\gamma = \gamma(n) \geq 1$, the decision problem γ-GapCVP is defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$, a target $\mathbf{t} \in \mathbb{R}^n$, and a number $d > 0$. The goal is to output yes if $\operatorname{dist}(\mathbf{t}, \mathcal{L}) < d$ and no if $\operatorname{dist}(\mathbf{t}, \mathcal{L}) \geq \gamma \cdot d$.

▶ **Definition 16.** For any parameter $\gamma = \gamma(n) \geq 1$, γ-uSVP (the Unique Shortest Vector Problem) is the search promise problem defined as follows: The input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ with $\lambda_2(\mathcal{L}) \geq \gamma(n) \cdot \lambda_1(\mathcal{L})$. The goal is to output a lattice vector $\mathbf{x}$ with $\|\mathbf{x}\| = \lambda_1(\mathcal{L})$.

2.4 Known results

We will need the following known reductions and hardness results.

▶ **Theorem 17** ([24]). *For any constant $\gamma \geq 1$, γ-GapSVP (and therefore γ-SVP) is NP-hard under randomized reductions.*

▶ **Theorem 18** ([19]). *For any $\gamma \geq 1$, there is an efficient dimension-preserving reduction from γ-GapSVP to γ-GapCVP (and from γ-SVP to γ-CVP).*

▶ **Theorem 19** ([29, Theorem 6.1]). *For any $1 \leq \gamma(n) \leq \mathrm{poly}(n)$, there is an efficient dimension-preserving reduction from γ-uSVP to γ-GapSVP.*

▶ **Theorem 20** ([33, Theorem 4.2]). *There is an efficient reduction from $\sqrt{n}$-CVP to $\sqrt{2}$-SVP. Furthermore, all of the oracle calls of the reduction are made in dimension $n+1$, where n is the input dimension.*

Reductions like that of Theorem 20 that increase the dimension by one are good enough for nearly all applications of perfectly dimension-preserving reductions. But, we can use a simple idea to convert Theorem 20 into a reduction that preserves the dimension exactly. (Micciancio uses essentially the same trick in the proof of [32, Corollary 7].)

▶ **Corollary 21.** *There is a dimension-preserving efficient reduction from $\sqrt{n}$-CVP to $\sqrt{2}$-SVP.*

Proof. On input $\mathbf{t} \in \mathbb{R}^n$ and a lattice $\mathcal{L} \subset \mathbb{R}^n$, the reduction first uses its SVP oracle to find a vector $\mathbf{b}_1^* \in \mathcal{L}^*$ in the dual with $0 < \|\mathbf{b}_1^*\| < 2\lambda_1(\mathcal{L}^*)$. Let $\mathbf{B}^* := (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ be a basis for $\mathcal{L}^*$, and let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be the associated primal basis. (Since $\mathbf{b}_1^*$ is a primitive lattice vector, it is always possible to find such a basis.) Let $\mathcal{L}' := \mathcal{L}(\mathbf{b}_2, \ldots, \mathbf{b}_n)$ be the lattice generated by $\mathbf{B}$ with the first basis vector removed. Finally, let $a := \langle \mathbf{b}_1^*, \mathbf{t} \rangle$. For $i = \lfloor a \rceil - n, \ldots, \lceil a \rceil + n$, the reduction runs the procedure from Theorem 20 on input $\mathbf{t} - i\mathbf{b}_1$ and $\mathcal{L}'$, receiving as output $\mathbf{y}_i \in \mathcal{L}'$. The reduction then simply outputs a closest vector to $\mathbf{t}$ amongst the vectors $\mathbf{y}_i + i\mathbf{b}_1 \in \mathcal{L}$.

It is clear that the reduction is efficient. Furthermore, note that the reduction only uses the procedure from Theorem 20 with $(n-1)$-dimensional input. (Formally, we must project $\mathcal{L}'$ and $\mathbf{t} - i\mathbf{b}_1$ onto $\mathrm{span}(\mathcal{L}') \cong \mathbb{R}^{n-1}$.) Since that procedure increases dimension by one, this new reduction preserves dimension. For correctness, we note that Corollary 7 implies that there is a closest vector to $\mathbf{t}$ in one of the lattice hyperplanes $\mathcal{L}' + i\mathbf{y}_i$. The result then follows from Theorem 20. ◀

2.5 A note on decision and estimation

Formally, we consider *gapped decision problems*, which take as input a number $d > 0$ and some additional input I and require us to output YES if $f(I) \leq d$ and NO if $f(I) > \gamma d$, where f is some function and γ is the approximation factor. (For example, I may be some representation of a lattice and $f(I)$ may be the length of the shortest vector in the lattice.) However, it is sometimes convenient to work with *estimation problems*, which take only I as input and ask for a numerical output $\tilde{d}$ with $f(I) \leq \tilde{d} \leq \gamma f(I)$.

For the specific problems that we consider (and most "sufficiently nice" problems), the estimation variants are equivalent to the gapped decision problems as long as the lattice is "represented reasonably" by the input. For example, if $f(I)$ can be represented as a string of length at most $\mathrm{poly}(|I|)$ (e.g., $f(I)$ might be a rational number with bounded numerator and denominator), then we can use binary search and a gapped decision oracle to estimate $f(I)$ efficiently. This is true, for example, whenever the input is interpreted as a list of vectors with rational coordinates, using the standard representation of rational numbers. (See [33] for a careful discussion of this and related issues in the context of lattice problems.) We therefore make no distinction between gapped decision problems and estimation problems in the sequel, without worrying about the specific form of our input, or more generally, the specific representation of numbers.

3 Reducing SVP to uSVP (and GapSVP) via sparsification

3.1 Sparsification

For a lattice $\mathcal{L} \subset \mathbb{R}^n$, we write $\mathcal{L}_{\text{prim}}$ for the set of all primitive vectors in $\mathcal{L}$, and $\xi(\mathcal{L}, r) := |\mathcal{L}_{\text{prim}} \cap rB_2^n|/2$ for the number of primitive lattice vectors contained in a (closed) ball of radius r around the origin (counting $\mathbf{x}$ and $-\mathbf{x}$ as a single vector). The following theorem from [42] shows that sparsification behaves nicely with respect to primitive vectors, which is enough for our use case.

▶ **Theorem 22** ([42, Theorem 4.1])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with basis $\mathbf{B}$, primitive lattice vectors $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathcal{L}_{\text{prim}}$ with $\mathbf{y}_i \neq \pm\mathbf{y}_0$ for all $i > 0$, and prime $p \geq 101$, if $\xi(\mathcal{L}, \|\mathbf{y}_i\|) \leq p/(20 \log p)$ for all i, then*

$$\frac{1}{p} - \frac{N}{p^2} \leq \Pr\left[\langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{y}_0\rangle \equiv 0 \bmod p \text{ and } \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{y}_i\rangle \not\equiv 0 \bmod p \ \forall i > 0\right] \leq \frac{1}{p} ,$$

where $\mathbf{z} \in \mathbb{Z}_p^n$ is chosen uniformly at random.

From this, we can immediately derive the following proposition, which is a slight variant of [42, Proposition 4.2].

▶ **Proposition 23.** *There is an efficient (randomized) algorithm that takes as input a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a prime $p \geq 101$ and outputs a full-rank sublattice $\mathcal{L}' \subseteq \mathcal{L}$ such that for any r_1, r_2, with $\lambda_1(\mathcal{L}) \leq r_1 \leq r_2 < p\lambda_1(\mathcal{L})$ and $\xi(\mathcal{L}, r_2) \leq p/(20 \log p)$, we have*

$$\Pr[\lambda_1(\mathcal{L}') \leq r_1 \text{ and } \lambda_2(\mathcal{L}') > r_2] \geq \frac{\xi(\mathcal{L}, r_1)}{p} \cdot \left(1 - \frac{\xi(\mathcal{L}, r_2)}{p}\right) .$$

Proof. The algorithm takes as input a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$. It then samples $\mathbf{z} \in \mathbb{Z}_p^n$ uniformly at random and outputs

$$\mathcal{L}' := \{\mathbf{y} \in \mathcal{L} \ : \ \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{y}\rangle \equiv 0 \bmod p\} .$$

Let $N := \xi(\mathcal{L}, r_2) \leq p/(20 \log p)$, and let $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathcal{L}$ be the N unique primitive vectors in $\mathcal{L}$ satisfying $\|\mathbf{y}_1\| \leq \cdots \leq \|\mathbf{y}_N\| \leq r_2$ (taking only one vector from each pair $\pm\mathbf{y}$). Note that we have $\lambda_1(\mathcal{L}') \leq r_1$ and $\lambda_2(\mathcal{L}') > r_2$ if and only if $\mathbf{y}_i \in \mathcal{L}'$ for some $i \leq \xi(\mathcal{L}, r_1)$ and $\mathbf{y}_j \notin \mathcal{L}'$ for all $j \neq i$. (Here, we have used the fact that $r_2 < p\lambda_1(\mathcal{L})$ to guarantee that vectors of the form $p\mathbf{y}_i \in \mathcal{L}'$ do not cause $\lambda_2(\mathcal{L}')$ to be less than r_2.)

Applying Theorem 22, we see that this happens with probability at least $1/p - (N-1)/p^2 > 1/p - N/p^2$ for any fixed i. The result follows by noting that these are disjoint events, so that the probability that at least one of these events occurs is the sum of their individual probabilities, which is at least

$$\xi(\mathcal{L}, r_1) \cdot \left(\frac{1}{p} - \frac{N}{p^2}\right) = \frac{\xi(\mathcal{L}, r_1)}{p} \cdot \left(1 - \frac{\xi(\mathcal{L}, r_2)}{p}\right) ,$$

as needed. ◀

3.2 The reduction

We now present the main step in our search-to-decision reduction for SVP.

▶ **Theorem 24.** *For any $\gamma = \gamma(n) \geq 1$ and $a = a(n) \geq \log(n+1)$, there is a dimension-preserving (randomized) reduction from $\gamma^{n/a}$-SVP to γ-uSVP that runs in time $2^{O(a)} \cdot \gamma^{O(n)}$.*

Proof. We may assume without loss of generality that $a \le n/2$, since the result is trivial for larger a. (There are known $2^{O(n)}$-time algorithms for SVP [6, 1].) We may also assume that $2^a > \gamma^n$, since this does not affect the asymptotic running time. Let $k := \lceil 4^{an/(n-a)} \rceil = 2^{O(a)}$.

On input a lattice $\mathcal{L} \subset \mathbb{R}^n$, the reduction does the following k times. For $i = 0, \ldots, \ell := \lfloor n/a \rfloor$, let p_i be a prime with $2k^{i+1} < p_i < 4k^{i+1}$. The reduction calls the procedure from Proposition 23 with input $\mathcal{L}$ and p_i, receiving as output $\mathcal{L}_i$. It then calls its uSVP oracle on each $\mathcal{L}_i$, receiving as output $\mathbf{x}_i$. Finally, it simply outputs a shortest non-zero $\mathbf{x}_i$.

It is clear that the reduction runs in time $\mathrm{poly}(n) \cdot k = 2^{O(a)}$, as needed.[5] For each i, let r_i be minimal such that $\xi(\mathcal{L}, r_i) \ge k^i$. In particular, $r_0 = \lambda_1(\mathcal{L})$. And, recalling the definition of ξ, we have

$$|\mathcal{L} \cap r_\ell B_2^n| > 2\xi(\mathcal{L}, r_\ell) \ge 2k^\ell > 2 \cdot (4^{an/(n-a)})^{n/a-1} = 2 \cdot 4^n \; .$$

So, applying Theorem 5, we have that $r_\ell / r_0 = r_\ell / \lambda_1(\mathcal{L}) > 2$.

Therefore, there exists an i such that $r_{i+1}/r_i > 2^{1/\ell} \ge 2^{a/n} > \gamma$. Let j be minimal such that $r_{j+1}/r_j > \gamma$. In particular, this means that $\xi(\mathcal{L}, \gamma r_j) < k^{j+1}$ and $\gamma r_j \le 2\gamma\lambda_1(\mathcal{L}) < p_j \lambda_1(\mathcal{L})$. So, we may apply Proposition 23 to obtain

$$\begin{aligned}\Pr[\lambda_1(\mathcal{L}_j) \le r_j \text{ and } \lambda_2(\mathcal{L}_j) > \gamma r_j] &\ge \frac{\xi(\mathcal{L}, r_j)}{p_j} - \frac{\xi(\mathcal{L}, r_j)\xi(\mathcal{L}, \gamma r_j)}{p_j^2} \\ &> \frac{k^j}{p_j} \cdot \Big(1 - \frac{k^{j+1}}{p_j}\Big) \\ &> \frac{1}{2k} \; .\end{aligned}$$

Therefore, after running the above procedure k times, the algorithm will output a non-zero vector of length at most r_i with at least some positive constant probability.

Finally, by the definition of r_j, we have $r_j/\lambda_1(\mathcal{L}) = r_j/r_0 \le \gamma^j \le \gamma^\ell \le \gamma^{n/a}$. Therefore, the algorithm outputs a $\gamma^{n/a}$-approximate shortest vector with at least constant probability, as needed. ◀

3.3 Corollaries

From this, we derive some immediate corollaries. The first is our main SVP result.

Proof of Theorem 1. We may assume without loss of generality that $\gamma \le 2$, since otherwise the result is trivial as there are known $2^{O(n)}$-time algorithms for SVP. Therefore, by Theorem 19, there is an efficient dimension-preserving reduction from γ-uSVP to γ-GapSVP. The result then follows from Theorem 24. ◀

By combining Theorem 24 with Corollary 21, we obtain the following reduction from $\sqrt{n}$-CVP to γ-uSVP (and therefore γ-GapSVP).

▶ **Corollary 25.** *For any $\gamma = \gamma(n) \ge 1$, there is a dimension-preserving (randomized) reduction from $\sqrt{n}$-CVP to γ-uSVP that runs in time* $\mathrm{poly}(n) \cdot \gamma^{O(n)}$.

Similarly, there is a dimension-preserving (randomized) reduction from $\sqrt{n}$-CVP to γ-GapSVP with the same running time.

[5] The reader might notice that the theorem quotes a running time of $2^{O(a)} \cdot \gamma^{O(n)}$ instead of just $2^{O(a)}$. Note that this looser bound on the running time is exactly what allowed us to assume $2^a > \gamma^n$ above. Equivalently, we could simply require $a > n \log \gamma$ in the theorem statement and achieve a running time of $2^{O(a)}$. But, we wish to avoid misunderstanding by explicitly stating that the running time is at least $\gamma^{O(n)}$ in the theorem statement.

Proof. Setting $a := 2n\log(\gamma) + \log(n+1)$ in Theorem 24 gives a reduction from $\sqrt{2}$-SVP to γ-uSVP with the claimed running time. The first result then follows from Corollary 21. The second result follows from Theorem 19. ◀

We also obtain an alternative proof of the hardness of $(1 + 1/\mathrm{poly}(n))$-uSVP, as originally shown by Aggarwal and Dubey [2].

▶ **Corollary 26.** *For any constant $\varepsilon > 0$, $(1 + 1/n^{\varepsilon})$-uSVP is NP-hard (under randomized reductions).*

Proof. For $\gamma \le 1 + O(\log n/n)$, taking $a := n\log(\gamma) + \log(n+1)$ in Theorem 24 gives a polynomial-time reduction from 2-SVP to γ-uSVP. It then follows from Theorem 17 that γ-uSVP is NP-hard (under randomized reductions).

The full result then follows by noting that there is a simple reduction from $(1 + 1/n^{\varepsilon})$-uSVP to $(1 + 1/n)$-uSVP for any constant $\varepsilon \in (0, 1)$. In particular, given input $\mathcal{L} \subset \mathbb{R}^n$ with basis $\mathbf{B} := (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, let $N := \lceil n^{1/\varepsilon} \rceil = \mathrm{poly}(n)$, and let $r := 3\|\mathbf{b}_1\| > 2\lambda_1(\mathcal{L})$. Let $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n, r\mathbf{e}_{n+1}, \ldots, r\mathbf{e}_N) \subset \mathbb{R}^N$ be the rank N lattice obtained by "adding $N - n$ perpendicular vectors of length r to $\mathcal{L}$." The result follows by noting that $N^{\varepsilon} \ge n$ so that $\mathcal{L}'$ is a valid instance of $(1 + 1/N^{\varepsilon})$-uSVP if $\mathcal{L}$ is a valid instance of $(1 + 1/n)$-uSVP, and the two instances have the same solution. ◀

Finally, we note that a reduction to GapSVP immediately implies a reduction to GapCVP, by Theorem 18. We will need this in the next section.

▶ **Corollary 27.** *For any $\gamma = \gamma(n) \ge 1$ and $a = a(n) \ge \log(n+1)$, there is a dimension-preserving (randomized) reduction from $\gamma^{n/a}$-SVP to γ-GapCVP that runs in time $2^{O(a)} \cdot \gamma^{O(n)}$.*

Proof. Combine Theorem 1 with Theorem 18. ◀

4 Reducing CVP to GapCVP

▶ **Theorem 28.** *For any $\gamma = \gamma(n) \ge 1$, $h = h(n) \ge 1$, and integer $\ell = \ell(n) \ge 1$, there is a (deterministic) algorithm with access to a γ-GapCVP oracle and a h-SVP oracle that solves $\gamma^{n/\ell}$-CVP in time $(\mathrm{poly}(n) \cdot h)^{\ell}$. Furthermore, the dimension of the algorithm's oracle calls never exceeds the dimension of the input lattice.*

Proof. We show how to handle the case $\ell = 1$ and then describe how to extend the result to arbitrary ℓ. On input a lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\mathbf{t} \in \mathbb{R}^n$, the algorithm behaves as follows. If $n = 1$, then it solves the CVP instance directly. Otherwise, it first uses its SVP oracle to find a dual vector $\mathbf{b}_1^* \in \mathcal{L}^*$ with $\|\mathbf{b}_1^*\| \le h \cdot \lambda_1(\mathcal{L}^*)$. Let $\mathbf{b}_2^*, \ldots, \mathbf{b}_n^* \in \mathcal{L}^*$ such that $(\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ is a basis of $\mathcal{L}^*$, and let $(\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathcal{L}$ be the associated basis of the primal. (This is always possible if $\mathbf{b}_1^*$ is primitive in $\mathcal{L}^*$. If $\mathbf{b}_1^*$ is not primitive, then we can simply replace it with a primitive vector that is a scalar multiple of $\mathbf{b}_1^*$.)

Next, let $a := \langle \mathbf{b}_1^*, \mathbf{t} \rangle$ and $\mathcal{L}' := \mathcal{L}(\mathbf{b}_2, \ldots, \mathbf{b}_n)$. Then, for $i = \lfloor a - h \cdot n \rfloor, \ldots, \lceil a + h \cdot n \rceil$, the algorithm uses its GapCVP oracle to compute d_i such that $\mathrm{dist}(\mathbf{t} - i \cdot \mathbf{b}_1, \mathcal{L}') \le d_i \le \gamma \cdot \mathrm{dist}(\mathbf{t} - i \cdot \mathbf{b}_1, \mathcal{L}')$. The algorithm then picks an index i such that d_i is minimal and calls itself recursively on input $\mathcal{L}'$ and $\mathbf{t} - i \cdot \mathbf{b}_1$, receiving as output $\mathbf{y} \in \mathcal{L}'$. Finally, it outputs $\mathbf{y} + i\mathbf{b}_1 \in \mathcal{L}$.

It is clear that the running time is as claimed. By Corollary 7, there must exist some j such that $\mathrm{dist}(\mathbf{t} - j \cdot \mathbf{b}_1, \mathcal{L}') = \mathrm{dist}(\mathbf{t}, \mathcal{L})$, so that $d_j \le \gamma\,\mathrm{dist}(\mathbf{t}, \mathcal{L})$. Therefore, $d_i \le \gamma \cdot \mathrm{dist}(\mathbf{t}, \mathcal{L})$, and $\mathrm{dist}(\mathbf{t} - i \cdot \mathbf{b}_1, \mathcal{L}') \le \gamma \cdot \mathrm{dist}(\mathbf{t}, \mathcal{L})$. The result then follows from induction on the dimension n.

To handle arbitrary $\ell \geq 1$, the algorithm simply tries all recursive paths up to depth ℓ and chooses the path that yields the lowest approximate distance according to its γ-GapCVP oracle. Note that there are at most $(2hn+2)^\ell = (\mathrm{poly}(n) \cdot h)^\ell$ such paths, so the running time is as claimed. ◀

We obtain our main CVP reduction by combining the above result with our SVP reduction.

Proof of Theorem 2. We may assume without loss of generality that ℓ is an integer.

We can instantiate the SVP oracle required in Theorem 28 above by using Corollary 27. In particular, taking $a := n\log\gamma + \log(n+1)$ in Corollary 25 gives a reduction from 2-SVP to γ-GapCVP that runs in time $\mathrm{poly}(n) \cdot \gamma^{O(n)}$. By using this reduction to instantiate the h-SVP oracle in Theorem 28 with $h = 2$, we get a dimension-preserving reduction from $\gamma^{n/\ell}$-CVP to γ-CVP that runs in time $n^{O(\ell)} \cdot \gamma^{O(n)}$, as needed. ◀

5 Deterministic reductions

We now show deterministic search-to-decision reductions for SVP and CVP that achieve significantly worse parameters. Both reductions use the same basic idea, which is essentially to "find the coordinates of a short (or close) lattice point bit-by-bit."

5.1 The deterministic CVP reduction

We present the CVP reduction first because it is simpler.

Proof of Theorem 4. We may assume without loss of generality that p is an integer and $\gamma^2 < p$, since the result is trivial for larger γ.

On input a lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\mathbf{t} \in \mathbb{R}^n$, the reduction behaves as follows. If $n = 1$, then it solves the CVP instance directly. Otherwise, it first uses the procedure from Theorem 9, to compute an LLL-reduced basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ for $\mathcal{L}$. It then finds the nth coordinate of a close lattice vector to $\mathbf{t}$ "in base p," as follows. Let $\mathbf{t}_0 = \mathbf{t}$, and let $\mathcal{L}_i := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}, p^i \cdot \mathbf{b}_n)$ for all i. For $i = 0, \ldots, \ell - 1$, with $\ell \geq 1$ to be set in the analysis, the reduction uses its γ-GapCVP oracle to compute $d_{i,0}, \ldots, d_{i,p-1}$ such that $\mathrm{dist}(\mathbf{t}_i - jp^i \cdot \mathbf{b}_n, \mathcal{L}_{i+1}) \leq d_{i,j} \leq \gamma\, \mathrm{dist}(\mathbf{t}_i - jp^i \cdot \mathbf{b}_n, \mathcal{L}_{i+1})$. It then sets $\mathbf{t}_{i+1} = \mathbf{t}_i - j \cdot p^i \cdot \mathbf{b}_n$, where j is chosen such that $d_{i,j}$ is minimal.

Let

$$\mathbf{t}' := \mathbf{t}_\ell - p^\ell \cdot \left\lfloor \frac{\langle \mathbf{t}_\ell, \widetilde{\mathbf{b}}_n \rangle}{p^\ell \cdot \|\widetilde{\mathbf{b}}_n\|^2} \right\rceil \cdot \mathbf{b}_n \; .$$

The reduction then calls itself recursively on input $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$ and $\mathbf{t}'$, receiving as output $\mathbf{y}' \in \mathcal{L}'$. Finally, the reduction outputs $\mathbf{y}' + \mathbf{t} - \mathbf{t}' \in \mathcal{L}$.

Take

$$\ell := \left\lceil \frac{n + \log n + 2}{2\log(p/\gamma)} \right\rceil = O(n/\log p) \; .$$

It is clear that the running time is as claimed. We first show by induction that $\mathrm{dist}(\mathbf{t}_i, \mathcal{L}_i) \leq \gamma^i \cdot \mathrm{dist}(\mathbf{t}, \mathcal{L})$. For $i = 0$, this is trivial. For any $i > 0$, let $\mathbf{x}$ be a closest vector in $\mathcal{L}_{i-1}$ to $\mathbf{t}_{i-1}$, and assume for induction that $\|\mathbf{x} - \mathbf{t}_{i-1}\| \leq \gamma^{i-1}\, \mathrm{dist}(\mathbf{t}, \mathcal{L})$. Note that we can write $\mathbf{x} = \mathbf{x}' + cp^{i-1} \cdot \mathbf{b}_n$, where $\mathbf{x}' \in \mathcal{L}_i$ and $c \in \{0, \ldots, p-1\}$. In particular,

$$d_{i,c} \leq \gamma\, \mathrm{dist}(\mathbf{t}_{i-1} - cp^{i-1} \cdot \mathbf{b}_n, \mathcal{L}_i) = \gamma\|\mathbf{x} - \mathbf{t}_{i-1}\| \leq \gamma^i\, \mathrm{dist}(\mathbf{t}, \mathcal{L}) \; .$$

It follows from the definition of $\mathbf{t}_i$ that $\operatorname{dist}(\mathbf{t}_i, \mathcal{L}_i) \le d_{i,c} \le \gamma^i \operatorname{dist}(\mathbf{t}, \mathcal{L})$, as needed.

We now wish to show that $\operatorname{dist}(\mathbf{t}', \mathcal{L}') = \operatorname{dist}(\mathbf{t}_\ell, \mathcal{L}_\ell)$. Suppose not. Then, clearly we have that $\operatorname{dist}(\mathbf{t}_\ell, \mathcal{L}_\ell) \ge p^\ell \|\widetilde{\mathbf{b}}_n\|/2$. (To see this, consider the "distance in the direction of $\widetilde{\mathbf{b}}_n$.") Combining this with the above, we have $\operatorname{dist}(\mathbf{t}, \mathcal{L}) \ge (p/\gamma)^\ell \cdot \|\widetilde{\mathbf{b}}_n\|/2 \ge \sqrt{n}2^{n/2} \cdot \|\widetilde{\mathbf{b}}_n\|$, contradicting Lemma 11.

Combining everything together, we see that $\operatorname{dist}(\mathbf{t}', \mathcal{L}') \le \gamma^\ell \cdot \operatorname{dist}(\mathbf{t}, \mathcal{L})$. Finally, we assume for induction that $\|\mathbf{y}' - \mathbf{t}'\| \le \gamma^{\ell \cdot (n-1)} \operatorname{dist}(\mathbf{t}', \mathcal{L}') \le \gamma^{\ell \cdot n} \operatorname{dist}(\mathbf{t}, \mathcal{L})$. It follows that $\|(\mathbf{y}' - \mathbf{t}' + \mathbf{t}) - \mathbf{t}\| = \|\mathbf{y}' - \mathbf{t}'\| \le \gamma^{\ell n} \operatorname{dist}(\mathbf{t}, \mathcal{L}) = \gamma^{O(n^2/\log p)} \operatorname{dist}(\mathbf{t}, \mathcal{L})$, as needed. ◀

5.2 The deterministic SVP reduction

The SVP reduction uses essentially the same idea, but it is a bit more technical because the GapSVP oracle is so much weaker. The reduction is very similar to the reduction from unique SVP to GapSVP in [29].

Proof of Theorem 3. We may assume without loss of generality that p is a prime and $\gamma^3 < p$, since the result is trivial for larger γ. On input a lattice $\mathcal{L} \subset \mathbb{R}^n$, the reduction behaves as follows. If $n = 1$, it solves the SVP instance directly. Otherwise, let $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be an LLL-reduced basis for $\mathcal{L}$ (which we can compute efficiently by Theorem 9).

Our goal is to compute a sequence of sublattices $\mathcal{L} = \mathcal{L}^{(0)} \subset \mathcal{L}^{(1)} \subset \cdots \subset \mathcal{L}^{(\ell)}$, with $\ell \ge 1$ to be set in the analysis, such that the index of $\mathcal{L}^{(i+1)}$ over $\mathcal{L}^{(i)}$ is p, and $\lambda_1(\mathcal{L}^{(i+1)}) \le \gamma \cdot \lambda_1(\mathcal{L}^{(i)})$. In particular, we will take $\mathcal{L}^{(i)} := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}, a_1^{(i)}\mathbf{b}_{n-1} + a_2^{(i)}\mathbf{b}_n, a_3^{(i)}\mathbf{b}_n)$ for some $a_1^{(i)}, a_2^{(i)}, a_3^{(i)}$, starting with $a_1^{(0)} := 1$, $a_2^{(0)} := 0$, and $a_3^{(0)} := 1$. To compute the remaining coefficients, the reduction behaves as follows for $i = 0, \ldots, \ell - 1$. For $j = 0, \ldots, p-1$, let $\mathcal{L}_{i,j} := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}, a_1^{(i)}\mathbf{b}_{n-1} + a_2^{(i)}\mathbf{b}_n + j a_3^{(i)}\mathbf{b}_n, p a_3^{(i)}\mathbf{b}_n)$ and let $\mathcal{L}_{i,p} := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}, p \cdot (a_1^{(i)}\mathbf{b}_{n-1} + a_2^{(i)}\mathbf{b}_n), a_3^{(i)}\mathbf{b}_n)$. For each j, the reduction uses its GapSVP oracle to compute $d_{i,j}$ such that $\lambda_1(\mathcal{L}_{i,j}) \le d_{i,j} \le \gamma\lambda_1(\mathcal{L}_{i,j})$. Let j such that $d_{i,j}$ is minimal. The reduction then sets the coefficients so that $\mathcal{L}^{(i+1)} := \mathcal{L}_{i,j}$.[6]

Let k_1 be the largest power of p that divides $a_1^{(\ell)}$, and let k_2 be the largest power of p that divides $a_3^{(\ell)}$. If $k_1 \ge k_2$, the reduction sets $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}, \mathbf{b}_n)$ to be "$\mathcal{L}$ without $\mathbf{b}_{n-1}$." Otherwise, it sets $\mathcal{L}' := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}, a_1^{(\ell)}\mathbf{b}_{n-1} + a_2^{(\ell)}\mathbf{b}_n)$ to be "$\mathcal{L}^{(\ell)}$ without $\mathbf{b}_n$." It then calls itself recursively on $\mathcal{L}'$ and returns the result.

Take

$$\ell := \left\lceil \frac{n+3}{\log(p/\gamma^2)} \right\rceil = O(n/\log p) \; .$$

The running time is clear. We first show that $\lambda_1(\mathcal{L}^{(i+1)}) \le \gamma \cdot \lambda_1(\mathcal{L}^{(i)})$ for all i. Indeed, let $\mathbf{v} \in \mathcal{L}^{(i)}$ such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L}^{(i)})$. As in the previous proof, it suffices to observe that $\mathbf{v} \in \mathcal{L}_{i,c}$ for some c, as this will imply that

$$\lambda_1(\mathcal{L}^{(i+1)}) \le \min_i d_{i,j} \le d_{i,c} \le \gamma\lambda_1(\mathcal{L}_{i,c}) = \gamma\|\mathbf{v}\| = \gamma\lambda_1(\mathcal{L}^{(i)}) \; ,$$

as needed. To see this, note that we can write $\mathbf{v} = \sum_{i=1}^{n-2} r_i \mathbf{b}_i + r_{n-1} \cdot (a_1^{(i)}\mathbf{b}_{n-1} + a_2^{(i)}\mathbf{b}_n) + r_n \cdot a_3^{(i)}\mathbf{b}_n$ where $r_i \in \mathbb{Z}$. If $r_{n-1} \equiv 0 \bmod p$, then clearly $\mathbf{v} \in \mathcal{L}_{i,p}$. Otherwise, there is a

[6] I.e., if $j < p$, the reduction sets $a_1^{(i+1)} := a_1^{(i)}$, $a_2^{(i+1)} := a_2^{(i)} + j a_3^{(i)}$, and $a_3^{(i+1)} := p a_3^{(i)}$. Otherwise, it sets $a_1^{(i+1)} := p a_1^{(i)}$, $a_2^{(i+1)} := a_2^{(i)}$, and $a_3^{(i+1)} := a_3^{(i)}$.

$c \in \{0, \ldots, p-1\}$ such that $cr_{n-1} \equiv r_n \bmod p$. Then,

$$\begin{aligned}\mathbf{v} &= \sum_{i=1}^{n-2} r_i \mathbf{b}_i + r_{n-1} \cdot (a_1^{(i)} \mathbf{b}_{n-1} + a_2^{(i)} \mathbf{b}_n) + r_n a_3^{(i)} \mathbf{b}_n \\ &= \sum_{i=1}^{n-2} r_i \mathbf{b}_i + r_{n-1} \cdot (a_1^{(i)} \mathbf{b}_{n-1} + a_2^{(i)} \mathbf{b}_n + c a_3^{(i)} \mathbf{b}_n) + (r_n - c r_{n-1}) a_3^{(i)} \mathbf{b}_n \; .\end{aligned}$$

Note that by definition $r_n - cr_{n-1} \equiv 0 \bmod p$, so it follows that $\mathbf{v} \in \mathcal{L}_{i,c}$, as needed.

In particular, $\lambda_1(\mathcal{L}^{(\ell)}) \leq \gamma^\ell \lambda_1(\mathcal{L})$. Now, we claim that $\lambda_1(\mathcal{L}') \leq \lambda_1(\mathcal{L}^{(\ell)})$. Note that any point $\mathbf{y} = \sum a_i \mathbf{b}_i \in \mathcal{L}^{(\ell)} \setminus \mathcal{L}'$ has either $|a_n| \geq p^{\max(k_1,k_2)} \geq p^{\ell/2}$ or $|a_{n-1}| \geq p^{\ell/2}$ (depending on whether or not $k_1 \geq k_2$). But, by Lemma 10, this implies that $\|\mathbf{y}\| \geq 2^{-n/2-1} \cdot p^{\ell/2} \cdot \lambda_1(\mathcal{L}) > \gamma^\ell \lambda_1(\mathcal{L})$. Therefore, any vector in $\mathcal{L}^{(\ell)}$ of length at most $\lambda_1(\mathcal{L}^{(\ell)}) \leq \gamma^\ell \cdot \lambda_1(\mathcal{L})$ must be in $\mathcal{L}'$, as needed.

Finally, as in the previous proof, we can show by a simple induction argument that the output vector has length at most $\gamma^{\ell(n-1)} \lambda_1(\mathcal{L}') \leq \gamma^{\ell n} \lambda_1(\mathcal{L}) = \gamma^{O(n^2/\log p)} \cdot \lambda_1(\mathcal{L})$, as needed. ◀

Acknowledgments. I would like to thank Divesh Aggarwal, Huck Bennett, Zvika Brakerski, Daniel Dadush, Daniele Micciancio, and Oded Regev, all of whom provided helpful commentary on earlier drafts of this paper. In particular, Divesh noted that our SVP reduction shows hardness of unique SVP (Corollary 26), and Daniele noted that it implied a reduction from $\sqrt{n}$-CVP to γ-GapSVP (Corollary 21). I also thank the anonymous reviewers for their helpful comments.

References

1 Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the Shortest Vector Problem in 2^n time via discrete Gaussian sampling. In *STOC*, 2015.

2 Divesh Aggarwal and Chandan Dubey. Improved hardness results for unique Shortest Vector Problem, 2015. URL: `http://eccc.hpi-web.de/report/2013/076/`.

3 Dorit Aharonov and Oded Regev. Lattice problems in NP intersect coNP. *Journal of the ACM*, 52(5):749–765, 2005. Preliminary version in FOCS'04.

4 Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, pages 99–108. ACM, 1996.

5 Miklós Ajtai. The Shortest Vector Problem in L2 is NP-hard for randomized reductions. In *STOC*, 1998. `doi:10.1145/276698.276705`.

6 Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.

7 L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. `doi:10.1007/BF02579403`.

8 Shi Bai, Weiqiang Wen, and Damien Stehlé. Improved reduction from the Bounded Distance Decoding problem to the unique Shortest Vector Problem in lattices. In *ICALP*, 2016.

9 W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993. `doi:10.1007/BF01445125`.

10 U. Betke, M. Henk, and J.M. Wills. Successive-minima-type inequalities. *Discrete & Computational Geometry*, 9(1):165–175, 1993. `doi:10.1007/BF02189316`.

11 Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE, 2011.

12 Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, pages 1–12, 2014.

13 Jin-Yi Cai and Ajay Nerurkar. Approximating the SVP to within a factor $(1+1/\dim^{\varepsilon})$ is NP-hard under randomized reductions. *Journal of Computer and System Sciences*, 59(2):221–239, 1999. doi:10.1006/jcss.1999.1649.

14 Daniel Dadush and Gabor Kun. Lattice sparsification and the approximate Closest Vector Problem. In *SODA*, 2013.

15 Daniel Dadush, Chris Peikert, and Santosh Vempala. Enumerative lattice algorithms in any norm via M-ellipsoid coverings. In *FOCS*, pages 580–589. IEEE, 2011.

16 Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. On the Closest Vector Problem with a distance guarantee. In *CCC*, pages 98–109, 2014. doi:10.1109/CCC.2014.18.

17 Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. ACM, New York, 2009.

18 Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

19 Oded Goldreich, Daniele Micciancio, Shmuel Safra, and Jean-Paul Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999. doi:10.1016/S0020-0190(99)00083-6.

20 Ishay Haviv and Oded Regev. Tensor-based hardness of the Shortest Vector Problem to within almost polynomial factors. *Theory of Computing*, 8(23):513–531, 2012. Preliminary version in STOC'07.

21 Antoine Joux and Jacques Stern. Lattice reduction: A toolbox for the cryptanalyst. *Journal of Cryptology*, 11(3):161–185, 1998.

22 G. A. Kabatjanskiĭ and V. I. Levenšteĭn. Bounds for packings on the sphere and in space. *Problemy Peredači Informacii*, 14(1):3–25, 1978.

23 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):pp. 415–440, 1987. URL: http://www.jstor.org/stable/3689974.

24 Subhash Khot. Hardness of approximating the Shortest Vector Problem in lattices. *Journal of the ACM*, 52(5):789–808, September 2005. Preliminary version in FOCS'04.

25 S. Ravi Kumar and D. Sivakumar. On the unique shortest lattice vector problem. *Theoretical Computer Science*, 255(1,Äì2):641–648, 2001. doi:10.1016/S0304-3975(00)00387-X.

26 A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982. doi:10.1007/BF01457454.

27 Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

28 Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio. On Bounded Distance Decoding for general lattices. In *RANDOM*, 2006.

29 Vadim Lyubashevsky and Daniele Micciancio. On Bounded Distance Decoding, unique shortest vectors, and the minimum distance problem. In *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594. Springer, 2009. doi:10.1007/978-3-642-03356-8_34.

30 Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and Learning with Errors over rings. In *EUROCRYPT*, 2010.

31 Daniele Micciancio. The Shortest Vector Problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, March 2001. Preliminary version in FOCS 1998.

32 Daniele Micciancio. Efficient reductions among lattice problems. In *SODA*, pages 84–93. ACM, New York, 2008.

33 Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.

34 Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

35 Phong Q Nguyen and Jacques Stern. The two faces of lattices in cryptology. In *Cryptography and lattices*, pages 146–180. Springer, 2001.

36 Andrew M Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and computational number theory*, 42:75–88, 1990.

37 Chris Peikert. Public-key cryptosystems from the worst-case Shortest Vector Problem. In *STOC*, pages 333–342. ACM, 2009.

38 Chris Peikert and Alon Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In *STOC*, 2007.

39 Xavier Pujol and Damien Stehlé. Solving the shortest lattice vector problem in time $2^{2.465n}$. *IACR Cryptology ePrint Archive*, 2009:605, 2009.

40 Oded Regev. On lattices, Learning with Errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):Art. 34, 40, 2009. doi:10.1145/1568318.1568324.

41 Noah Stephens-Davidowitz. Dimension-preserving reductions between lattice problems, 2015. URL: http://noahsd.com/latticeproblems.pdf.

42 Noah Stephens-Davidowitz. Discrete Gaussian sampling reduces to CVP and SVP. In *SODA*, 2016.

Proving Weak Approximability Without Algorithms*

Ridwan Syed[1] **and Madhur Tulsiani**[2]

1 **University of Chicago, Chicago, IL, USA**
rsyed@uchicago.edu

2 **Toyota Technological Institute at Chicago, Chicago, IL, USA**
madhurt@ttic.edu

Abstract

A predicate $f : \{-1,1\}^k \mapsto \{0,1\}$ with $\rho(f) = \mathbb{E}_{x\in\{-1,1\}^k}[f(x)]$ is said to be *strongly approximation resistant* if, for every $\varepsilon > 0$, given a near-satisfiable instance of MAX k-CSP(f), it is hard to find an assignment such that the fraction of constraints satisfied is outside the range $[\rho(f) - \varepsilon, \rho(f) + \varepsilon]$. A predicate which is not strongly approximation resistant is known as *weakly approximable.*

We give a new method for proving the weak approximability of predicates, using a simple SDP relaxation, without designing and analyzing new rounding algorithms for each predicate. Instead, we use the recent characterization of strong approximation resistance by Khot et. al [13], and show how to prove that for a given predicate f, certain necessary conditions for strong resistance derived from their characterization, are violated. By their result, this implies the existence of a good rounding algorithm, proving weak approximability.

We show how this method can be used to obtain simple proofs of (weak approximability analogues of) various known results on approximability, as well as new results on weak approximability of symmetric predicates.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases approximability, constraint satisfaction problems, approximation resistance, linear programming, semidefinite programming

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.20

1 Introduction

Constraint Satisfaction Problems (CSPs) are some of the most basic problems in the study of approximation algorithms and inapproximability. The problem MAX k-CSP(f) is characterized by a Boolean predicate $f : \{-1,1\}^k \to \{0,1\}$. An instance of the problem is described by (say) n variables $x_1, \ldots, x_n$ taking values in $\{-1,1\}$, and a set of (say) m constraints where each constraint C_i is of the form $C_i \equiv f(x_{i_1} \cdot b_{i_1}, \ldots, x_{i_k} \cdot b_{i_k})$ for some $b_{i_1}, \ldots, b_{i_k} \in \{-1,1\}$. An assignment $\sigma : \{x_1, \ldots, x_n\} \to \{-1,1\}$ is said to satisfy the constraint C_i if $f(\sigma(x_{i_1}) \cdot b_{i_1}, \ldots, \sigma(x_{i_k}) \cdot b_{i_k}) = 1$. Given an instance of the problem, the goal is to find an assignment satisfying the maximum possible number of constraints. For a given instance Φ, we denote the *fraction* of constraints satisfied by the optimal assignment as OPT(Φ). An algorithm is said to achieve an approximation factor α if it always produces an assignment satisfying at least $\alpha \cdot$ OPT(Φ) fraction of constraints.

* Research supported by NSF Career Award CCF-1254044.

Given an instance of MAX k-CSP, a trivial algorithm is to assign independently to each variable x_i a random value in $\{-1, 1\}$. This satisfies a fraction of constraints concentrated around the quantity $\rho(f) = \mathbb{E}_{x\in\{-1,1\}^k}[f(x)]$. A predicate for which this approximation is best possible i.e. for every $\varepsilon > 0$, given an instance with $\mathsf{OPT} \geq 1-\varepsilon$ it is (NP/UG-) hard find an assignment satisfying $\rho(f)+\varepsilon$ fraction of constraints, is known as approximation resistant. An even stronger notion of hardness, which was implicit in the literature on hardness of approximation, and was explicitly defined by Khot et. al. [13], is known as strong approximation resistance. A predicate is said to be strongly approximation resistant, if it is hard to find an assignment which *significantly deviates* from a random assignment i.e. for every $\varepsilon > 0$, given an instance with $\mathsf{OPT}(\Phi) \geq 1-\varepsilon$, it is (NP/UG-) hard to find an assignment satisfying a fraction of constraints outside the interval $[\rho(f)-\varepsilon, \rho(f)+\varepsilon]$. A predicate which is not approximation resistant is known as approximable, and one which is not strongly resistant is known as weakly approximable. Note that for an odd predicate i.e.a predicate satisfying $f(x) = 1 - f(-x)\ \forall x$, the notions of approximability and weak approximability are equivalent.

The notion of approximation resistance has been extensively studied, starting from the celebrated result of Håstad [9] showing that MAX 3-SAT and MAX 3-XOR are approximation resistant. Since then, many predicates have been shown to be approximation resistant (see e.g. [7, 16, 11, 4], all proving NP-hardness). Recently, a remarkable result by Chan [2] proved the approximation resistance of the Hypergraph Linearity Predicate (he shows NP-hardness whereas UG-hardness was shown earlier in [17]).

Assuming the Unique Games Conjecture (UGC) of Khot [12], Austrin and Mossel [1] show that any predicate f for which $f^{-1}(1)$ supports a balanced and pairwise independent distribution on $\{-1, 1\}^k$, is approximation resistant. In addition to the above results, a very general result by Raghavendra [15] also shows that assuming the UGC, the best possible approximation for any problem of the form MAX k-CSP(f) (for any f) can be obtained by a certain Semidefinite Programming (SDP) relaxation (see Fig. 1) known as the basic SDP. Thus, assuming the UGC, a predicate is approximation resistant if and only if one cannot do better than the trivial algorithm, using the basic SDP. All of the above results on approximation resistance in fact prove that the predicates in question are *strongly resistant*[1].

For the case of approximability, it follows from the algorithm of Goemans and Williamson [5] (and was shown by Håstad for every alphabet size [10]) that every predicate on 2 inputs is approximable. A classification for all predicates of arity 3 follows from the work of Zwick (see [19, 18]), and a large number of predicates of arity 4 were classified by Hast [8]. Hast also gave a general sufficient condition (discussed later) for a predicate of arity k to be approximable. He provided a rounding algorithm for an SDP relaxation, which achieves a good approximation assuming the above condition.

For predicates with large arity, approximability results are known for various special cases. The case when f is the sign of a linear polynomial in the variables, was studied by Cherghachi et. al.[3]. They defined a special subclass, which they called "Chow-robust" predicates, for which approximability follows from the sufficient condition of Hast. They studied the approximability curve for these predicates, adapting Hast's algorithm to obtain the best possible approximation. Austrin et. al.[14] proved approximability for the case when f is the sign of a quadratic polynomial which is symmetric in all the variables (with constant term 0), again by using the algorithm of Hast (for which they gave a simpler analysis). They also studied a new predicate, known as the "Monarchy" predicate, for which approximability

[1] To the best of our knowledge, this is true for all known results proving approximation resistance.

does not follow from Hast's algorithm. They gave a new algorithm to obtain a non-trivial approximation for the Monarchy predicate.

A related result is the study of approximation resistance for *symmetric predicates* in k variables, by Guruswami and Lee [6]. They study whether the sufficient condition of Austrin and Mossel [1] of $f^{-1}(1)$ supporting a balanced pairwise independent distribution, is also necessary for the case of symmetric predicates[2]. They show this to be the case when f is a *even* symmetric predicate, or corresponds to an *interval* of values for $\sum_{i=1}^{k} x_i$. Since the Austrin-Mossel condition is sufficient for approximation resistance, showing that it is necessary corresponds to proving an approximability result. As before, this is also proved using the condition (and algorithm) by Hast [8].

A characterization of strong approximation resistance

The starting point for our work is a characterization of strong approximation resistance, recently given by Khot et. al.[13]. Their characterization is in terms of a polytope $\mathbb{C}(f)$ associated with the predicate f. For a distribution μ supported on a subset of $f^{-1}(1)$, let $\zeta = \zeta(\mu)$ denote the $(k+1) \times (k+1)$ moment matrix with $\zeta(i,j) = \mathbb{E}_{x\sim\mu}[x_i x_j]$ and $\zeta(0,i) = \zeta(i,0) = \mathbb{E}_{x\sim\mu}[x_i]$. Then $\mathbb{C}(f)$ is defined as the convex polytope

$$\mathcal{C}(f) = \{\zeta(\mu) \ : \ supp(\mu) \subseteq f^{-1}(1)\}\,.$$

The condition of Khot et. al.says that a predicate f is strongly approximation resistant *if and only if* there exists a probability measure Λ on $\mathbb{C}(f)$, satisfying certain symmetry properties. These properties amount to saying for a set of k linear transformations $L_1, \ldots, L_k$ (with L_t depending on Fourier coefficients for sets of size t), we get $L_t(\Lambda) \equiv 0$ for all $t \in [k]$. We refer to such a measure Λ as a vanishing measure.

The results of Khot et. al.in fact characterize approximability with respect to the basic SDP relaxation. They show that if a vanishing measure exists, then for every $\varepsilon > 0$, there exist instances such that the value of the SDP relaxation is at least $1-\varepsilon$, but for every assignment to the variables, the fraction of constraints satisfied is in the interval $[\rho(f)-\varepsilon, \rho(f)+\varepsilon]$. By the results of Raghavendra [15], this shows that the predicate is strongly approximation resistant (assuming the UGC). Conversely, they also show that if such a measure does not exist, then there exists a (randomized) rounding algorithm for the basic SDP, which given an SDP solution with value at least $1-\varepsilon$, produces an assignment whose value deviates from $\rho(f)$ by at least ε (in expectation).

Our results

The goal of this work is to show how various results on (weak) approximability can be proved using the characterization of Khot et. al., without designing a new rounding algorithm in each case. Given their result, it suffices to show the nonexistence of a vanishing measure, to show the existence of a good rounding algorithm. For a variety of cases including the condition of Hast [8] and the Monarchy predicate, we show that the showing the nonexistence of a vanishing measure turns out to be much simpler to prove. We also derive some new results on weak approximability results of symmetric predicates, as described below.

We prove the following results corresponding to the approximability condition of Hast. We note that our proof only gives weak approximability, while the proofs by Hast [8] and

[2] Guruswami and Lee consider CSPs both with and without negation. However, we only discuss the former here.

Austrin et. al.[14] based on a rounding algorithm, give approximability under the same condition on f. The result below also gives weak approximability for any results based on Hast's condition.

▶ **Theorem 1.** *Let $f : \{-1,1\} \to \{0,1\}$ be a predicate. Suppose there exists $\eta \in \mathbb{R}$, such that*

$$\frac{2\eta}{\sqrt{2\pi}} \cdot \sum_i \hat{f}(\{i\}) \cdot x_i + \frac{2}{\pi} \cdot \sum_{i<j} \hat{f}(\{i,j\}) \cdot x_i x_j \ > \ 0$$

for all $x \in f^{-1}(x)$. Then f is weakly approximable.

For the Monarchy predicate, we prove the following result (proved by Austrin et. al.[14] using a different rounding algorithm than the one used for the above result)

▶ **Theorem 2.** *Let f be the Monarchy predicate defined as*

$$f(x) \ := \ \frac{1 + sgn\left((k-2) \cdot x_1 + x_2 + \cdots + x_k\right)}{2}.$$

Then f is approximable using the basic SDP.

Note that since Monarchy is an odd predicate, the notions of approximability and weak approximability are equivalent in this case. Finally, we prove that for a symmetric predicate f with non-zero Fourier mass on sets of size 1 and 2, the condition of Austrin and Mossel is tight for strong approximation resistance i.e.f is strongly approximation resistant if and only if $f^{-1}(1)$ supports a balanced pairwise independent distribution. As discussed before, the condition is known to be sufficient for strong approximation resistance, and thus showing that it is necessary is a result about (weak) approximability.

▶ **Theorem 3.** *Let $f : \{-1,1\}^k \to \{0,1\}$ be a symmetric predicate such that either $\hat{f}(\{1\}) = \hat{f}(\{2\}) = 0$, or $\hat{f}(\{1\}) \neq 0$ and $\hat{f}(\{1,2\}) \neq 0$. Then f is strongly approximation resistant if and only if $f^{-1}(1)$ supports a balanced pairwise independent distribution.*

We remark that in the first case of the above theorem, the uniform distribution on $f^{-1}(1)$ is balanced and pairwise independent. Hence, the interesting part of the result is in the case Fourier coefficients are non-zero at both the levels.

We conclude this section with two brief remarks on our techniques and the issue of approximability vs. weak approximability. First, note that the idea of proving a nonexistence result (for a vanishing measure) instead of an existence result (for an algorithm) seems counterintuitive, since we switch from an existential quantifier to a universal one. However, we in fact show the non-existence of a vanishing measure by showing the *existence* of a function h and a $t \in [k]$ such that $\int h \cdot L_t(\Lambda) \neq 0$ (and hence $L_t(\Lambda) \not\equiv 0$ showing that Λ is not a vanishing measure). The function h turns out to be a simpler "core" object which encodes all the required information for a rounding algorithm, but is easier to argue about We characterize the class of functions h which we search over in Section 3, providing a single framework to capturing various known results.

Secondly, we remark that results in this work only prove weak approximability, and do not necessarily find the best approximation threshold for a problem. However, in many cases, the reason for proving approximability, is in fact to rule out approximation resistance. In such cases, it also seems interesting to rule out strong approximation resistance (i.e.prove weak approximability) since the known techniques for proving approximation resistance, seem to also prove strong resistance.

2 Preliminaries

2.1 Constraint Satisfaction Problems

▶ **Definition 4.** For a predicate $f : \{-1,1\}^k \to \{0,1\}$, an instance Φ of MAX k-CSP (f) consists of a set of variables $\{x_1,\ldots,x_n\}$ and a set of constraints $C_1,\ldots,C_m$ where each constraint C_i is over a k-tuple of variables $\{x_{i_1},\ldots,x_{i_k}\}$ and is of the form

$$C_i \;\equiv\; f(x_{i_1}\cdot b_{i_1},\ldots,x_{i_k}\cdot b_{i_k})$$

for some $b_{i_1},\ldots,b_{i_k} \in \{-1,1\}$. For an assignment $\sigma : \{x_1,\ldots,x_n\} \mapsto \{-1,1\}$, let $\mathsf{sat}(\sigma)$ denote the fraction of constraints satisfied by σ. The maximum fraction of constraints that can be simultaneously satisfied is denoted by $\mathsf{OPT}(\Phi)$, i.e.

$$\mathsf{OPT}(\Phi) = \max_{\sigma:\{x_1,\ldots,x_n\}\mapsto\{-1,1\}} \mathsf{sat}(\sigma).$$

The density of the predicate is $\rho(f) = \frac{|f^{-1}(1)|}{2^k}$.

▶ **Definition 5.** A predicate $f : \{-1,1\}^k \to \{0,1\}$ is called approximable if there exists a constant $\varepsilon > 0$ and a polynomial time algorithm, possibly randomized, that given an $(1-\varepsilon)$-satisfiable instance of MAX k-CSP (f), outputs an assignment A such that $\mathbb{E}_A\left[\,\mathsf{sat}(A)\,\right] \geq \rho(f)+\varepsilon$. Here the expectation is over the randomness used by the algorithm. The predicate is called weakly approximable if the output of the algorithm deviates from $\rho(f)$ in expectation, i.e. $\mathbb{E}_A\left[\,|\mathsf{sat}(A)-\rho(f)|\,\right] \geq \varepsilon$.

Note that the two notions are equivalent for an odd predicate satisfying $f(x) = 1 - f(-x)$ for all $x \in \{-1,1\}^k$

A predicate that is not approximable is said to be approximation resistant and a predicate that is not weakly approximable is said to be strongly approximation resistant. However, since these conditions require the *non-existence* of algorithms, one can only define them under certain conjectures such as the Unique Games conjecture of Khot [12] (and $P \neq NP$), or for a specific family of algorithms.

It follows from a result of Raghavendra [15] that approximation resistance with respect to a specific algorithm given by a basic SDP relaxation, discussed in the next section, is equivalent to approximation resistance assuming the UGC. It was observed in [13] that this is also true for strong resistance. Thus, we will limit ourselves to discussion of resistance with respect to the basic SDP relaxation. Since the goal here is to prove approximability, we in fact prove that the problems in question are approximable using the basic SDP.

2.2 Fourier Analysis

Recall that a function $f : \{-1,1\}^k \to \mathbb{R}$ can be written as

$$f \;=\; \sum_{S\subseteq[k]} \hat{f}(S)\cdot\chi_S\,,$$

where the functions $\chi_S(x) = \prod_{i\in S} x_i$ form an orthonormal basis for the space of functions $f : \{-1,1\}^k \to \mathbb{R}$ under the inner product $\langle f,g\rangle = \mathbb{E}_{x\in\{-1,1\}^k}\left[f(x)g(x)\right]$. The coefficients $\hat{f}(S)$ are known as Fourier coefficients and can be computed as

$$\hat{f}(S) \;=\; \langle f,\chi_S\rangle \;=\; \mathbb{E}\left[\prod_{i\in S} x_i\cdot f(x)\right].$$

$$
\begin{array}{rcll}
\text{maximize} & & \mathop{\mathbb{E}}_{C\in\Phi}\left[\displaystyle\sum_{\alpha\in\{-1,1\}^k} f(\alpha\cdot b_C)\cdot x_{(S_C,\alpha)}\right] & \\
\text{subject to} & & & \\
\left\langle \mathbf{v}_{(i,1)}, \mathbf{v}_{(i,-1)} \right\rangle & = & 0 & \forall i\in[n] \\
\mathbf{v}_{(i,1)} + \mathbf{v}_{(i,-1)} & = & \mathbf{v}_{(\emptyset,\emptyset)} & \forall i\in[n] \\
\left\| \mathbf{v}_{(\emptyset,\emptyset)} \right\|^2 & = & 1 & \\
\displaystyle\sum_{\substack{\alpha\in\{-1,1\}^{S_C}\\ \alpha(i_1)=b_1,\alpha(i_2)=b_2}} x_{(S_C,\alpha)} & = & \left\langle \mathbf{v}_{(i_1,b_1)}, \mathbf{v}_{(i_2,b_2)} \right\rangle & \forall C\in\Phi, i_1\neq i_2\in S_C, b_1,b_2\in\{-1,1\} \\
x_{(S_C,\alpha)} & \geq & 0 & \forall C\in\Phi,\ \forall\alpha\in\{-1,1\}^{S_C}
\end{array}
$$

Figure 1 Basic Relaxation for MAX k-CSP (f).

2.3 The Basic SDP Relaxation for CSPs

We present below the basic SDP relaxation considered by Raghavendra [15]. The relaxation is includes non-negative variables $x_{(S_C,\alpha)}$ are included for sets S_C corresponding to the set of CSP variables for some constraint C, and an assignment $\alpha\in\{-1,1\}^{S_C}$. The variables $\left\{x_{(S_C,\alpha)}\right\}_{\alpha\in\{-1,1\}^{S_C}}$ add up to 1, thus defining a distribution on the assignments to the CSP variables in the set S_C.

The relaxation also has vectors $\mathbf{v}_{(i,b)}$ for each $i\in[n]$ and $b\in\{-1,1\}$, such that the inner products $\left\langle \mathbf{v}_{(i_1,b_1)}, \mathbf{v}_{(i_2,b_2)} \right\rangle$ correspond to the probability that $x_{i_1}=b_1$ and $x_{i_2}=b_2$. The relaxation (after a minor rewriting) is shown in Fig. 1.

For an SDP relaxation of MAX k-CSP, and for a given instance Φ of the problem, we denote by FRAC(Φ) the SDP (fractional) optimum. For the particular instance Φ, the integrality gap is defined as FRAC(Φ)/OPT(Φ). The integrality gap of the relaxation is the supremum of integrality gaps over all instances. The integrality gap thus defined is in terms of a ratio whereas we are concerned with the specific *gap* location $1-o(1)$ versus $\rho(f)+o(1)$ and also with the *strong* integrality gap as defined below.

▶ **Definition 6.** Let $\varepsilon>0$ be a constant. A relaxation is said to have a $(1-\varepsilon,\rho(f)+\varepsilon)$-integrality gap if there exists a CSP instance Φ such that FRAC$(\Phi)\geq 1-\varepsilon$ and OPT$(\Phi)\leq \rho(f)+\varepsilon$.

The relaxation is said to have a strong $(1-\varepsilon,\rho(f)\pm\varepsilon)$-integrality gap if there exists a CSP instance Φ such that FRAC$(\Phi)\geq 1-\varepsilon$ and for every assignment σ to the instance, $|\mathsf{sat}(\sigma)-\rho(f)|\leq\varepsilon$.

It was shown by Raghavendra [15] that the integrality gap for the basic relaxation as in Fig. 1 implies a UG-hardness result. It was observed by Khot et. al.[13] that this also holds for strong integrality gaps.

▶ **Theorem 7** ([15]). *If the basic SDP in Fig. 1 has a $(1-\varepsilon,\rho(f)+\varepsilon)$-integrality gap for every $\varepsilon>0$, then f is approximation resistant assuming the UGC. Moreover, if the SDP has a strong $(1-\varepsilon,\rho(f)\pm\varepsilon)$-gap for every $\varepsilon>0$, then f is strongly approximation resistant (assuming the UGC).*

2.4 Approximation Resistance Characterization

In this section, we briefly review the characterization of strong approximation resistance by Khot et. al.[13].

Let μ be a probability distribution over $\{-1,1\}^k$. Then the symmetric matrix of first and second moments $\zeta(\mu)$ is defined as follows

$$\zeta(\mu) = \begin{bmatrix} 1 & \mathbb{E}[x_1] & \mathbb{E}[x_2] & \cdots & \mathbb{E}[x_k] \\ \mathbb{E}[x_1] & 1 & \mathbb{E}[x_1x_2] & \cdots & \mathbb{E}[x_1x_k] \\ \mathbb{E}[x_2] & \mathbb{E}[x_1x_2] & 1 & \cdots & \mathbb{E}[x_2x_k] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}[x_k] & \mathbb{E}[x_1x_k] & \mathbb{E}[x_2x_k] & \cdots & 1 \end{bmatrix}$$

with $\mathbb{E}[x_i]$ in the $(0,i)$ entry, and $\mathbb{E}[x_ix_j]$ in the (i,j) entry. All expectations above are with respect to the distribution μ. The characterization of Khot et. al.is in terms of measures on the convex polytope

$$\mathcal{C}(f) = \{\zeta(\mu) \ \mid \ supp(\mu) \subseteq f^{-1}(1)\}\,.$$

To describe the characterization, we first consider three ways of transforming such a matrix ζ. All transformations preserve the symmetry of ζ.

- Projection to a subset S: Fix a nonempty $S \subset [k]$. Then ζ_S is the $|S|+1$ by $|S|+1$ principal submatrix obtained by restricting to rows and columns in $\{0\} \cup S$.
- Permuting rows/columns: Fix a permutation $\pi : S \to S$. Then, $\zeta_{S,\pi}$ is the $|S|+1$ by $|S|+1$ matrix obtained by permuting the rows and columns of ζ_S corresponding to S, according to π^{-1}, i.e.

$$\zeta_{S,\pi}(i,j) = \zeta_S(\pi(i),\pi(j)) \quad \forall i,j \in S \quad \text{and} \quad \zeta_{S,\pi}(i,0) = \zeta_{S,\pi}(0,i) = \zeta_S(0,\pi(i))\,.$$

- Applying a vector of signs: Fix $b \in \{-1,1\}^S$. Then, $\zeta_{S,\pi,b}$ is the $|S|+1$ by $|S|+1$ matrix obtained by taking the entry-wise product of $\zeta_{S,\pi,b}$ and $(1b)(1b)^T$, i.e.

$$\zeta_{S,\pi,b}(i,j) = b_ib_j \cdot \zeta_{S,\pi}(i,j) \quad \forall i,j \in S \quad \text{and} \quad \zeta_{S,\pi,b}(i,0) = \zeta_{S,\pi,b}(0,i) = b_i \cdot \zeta_{S,\pi}(0,i)\,.$$

Now, let Λ be a probability measure over $\mathcal{C}(f)$. Fix, nonempty $S \subset [k]$, $\pi : S \to S$, and $b \in \{-1,1\}^S$. We define the transformed measure $\Lambda_{S,\pi,b}$ (over $|S|+1$ by $|S|+1$ matrices) defined as

$$\Lambda_{S,\pi,b}(M) \ := \ \Lambda\left(\{\zeta \in \mathbb{C}(f) \ \mid \ \zeta_{S,\pi,b} = M\}\right)\,.$$

We now state the characterization of strong approximation resistance in terms of the basic SDP relaxation in Fig. 1.

▶ **Theorem 8** ([13]). *A given predicate $f : \{-1,1\}^k \to \{0,1\}$ is strongly approximation resistant for the basic SDP relaxation if and only if there exists a probability measure Λ supported on $\mathcal{C}(f)$ such that for all $t \in [k]$ the following function on matrices M is identically 0:*

$$\Lambda^{(t)}(M) \ = \ \sum_{|S|=t} \ \sum_{\pi : S \to S} \ \sum_{b \in \{-1,1\}^S} \Lambda_{S,\pi,b}(M) \cdot \hat{f}(S) \cdot \prod_{i \in S} b_i$$

Such a probability measure Λ is called a vanishing measure*.*

3 Weak Approximability of Predicates

We first derive a necessary condition for strong approximation resistance, using the characterization in Theorem 8. We will then derive various approximability results by showing that the necessary condition is violated.

Suppose $f : \{-1,1\}^k \to \{0,1\}$ is strongly approximation resistant. Theorem 8 implies that there exists a measure Λ supported on the convex body $\mathcal{C}(f)$ such that for all $t \in [k]$:

$$\Lambda^{(t)}(M) = \sum_{|S|=t} \sum_{\pi:S\to S} \sum_{b\in\{-1,1\}^{|S|}} \Lambda_{S,\pi,b}(M) \cdot \hat{f}(S) \cdot \prod_{i\in S} b_i$$

is an identically zero function. Then, for any function h we have

$$\int h(M) \cdot \Lambda^{(t)}(M) \ = 0 \, .$$

We use this to derive a necessary condition. For $t \in [k]$, let $h : [-1,1]^{(t+1)\times(t+1)} \to \mathbb{R}$ be a function on $(t+1)\times(t+1)$ matrices. We will consider matrices of the form $\zeta_{S,\pi,b}$ where $|S| = t$, $\pi : S \to S$ and $b \in \{-1,1\}^S$. We call h an odd symmetric function if for all $\zeta \in [-1,1]^{(t+1)\times(t+1)}$ with $\zeta(i,i) = 1$, all $\pi : [t] \to [t]$ and $b \in \{-1,1\}^t$, we have

$$h(\zeta_{\pi,b}) \ = \ \left(\prod_{i\in[t]} b_i\right) \cdot h(\zeta) \, .$$

Note that the permutations π only permute rows and columns $1, \dots, t$ but do not move the 0^{th} row or column (although the entries in the 0^{th} or column may be permuted). We now state our necessary condition for strong approximation resistance.

▶ **Lemma 9.** *Let $f : \{-1,1\}^k \to \{0,1\}$ be a predicate and let Λ be a vanishing measure on $\mathbb{C}(f)$ satisfying the condition in Theorem 8 for all $t \in [k]$. Let $h : [-1,1]^{(t+1)\times(t+1)} \to \mathbb{R}$ be an odd symmetric function. Then,*

$$\mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[\sum_{|S|=t} \hat{f}(S) \cdot h\left(\zeta_S\right)\right] \ = \ 0 \, .$$

Proof. The proof is a simple consequence of Theorem 8. Let M be a $(t+1)\times(t+1)$ matrix. Since Λ is a vanishing measure, we know that the signed measure $\Lambda^{(t)}$ should be identically zero. Thus, we have

$$\int \Lambda^{(t)}(M) \cdot h(M) \ = \ \int \left(\sum_{|S|=t} \ \sum_{\pi:S\to S} \ \sum_{b\in\{-1,1\}^S} \hat{f}(S) \cdot \prod_{i\in S} b_i \cdot \Lambda_{S,\pi,b}(M)\right) \cdot h(M) \ = \ 0 \, .$$

From the definition of $\Lambda_{S,\pi,b}$, we know that

$$\int \Lambda_{S,\pi,b}(M) \cdot h(M) \ = \ \mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[h\left(\zeta_{S,\pi,b}\right)\right] \, .$$

Thus, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[\sum_{|S|=t}\ \sum_{\pi:S\to S}\ \sum_{b\in\{-1,1\}^S} \hat{f}(S)\cdot\prod_{i\in S} b_i\cdot h\left(\zeta_{S,\pi,b}\right)\right] \;=\; 0\\
\Rightarrow\;&\mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[\sum_{|S|=t}\ \sum_{\pi:S\to S}\ \sum_{b\in\{-1,1\}^S} \hat{f}(S)\cdot h\left(\zeta_S\right)\right] \;=\; 0\\
\Rightarrow\;&\mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[\sum_{|S|=t} \hat{f}(S)\cdot h\left(\zeta_S\right)\right] \;=\; 0\,,
\end{aligned}$$

where the first implication uses the fact that h is an odd symmetric function. ◀

▶ Remark. The restriction to odd symmetric functions in the above lemma is actually without loss of generality. Starting from an arbitrary function g, we would get

$$\mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[\sum_{|S|=t}\ \sum_{\pi:S\to S}\ \sum_{b\in\{-1,1\}^S} \hat{f}(S)\cdot\prod_{i\in S} b_i\cdot g\left(\zeta_{S,\pi,b}\right)\right] \;=\; 0\,,$$

where

$$h\left(\zeta_S\right) \;=\; \sum_{\pi:S\to S}\ \sum_{b\in\{-1,1\}^S}\ \prod_{i\in S} b_i\cdot g\left(\zeta_{S,\pi,b}\right)$$

is an odd symmetric function of ζ_S.

We shall use Lemma 9 with different functions h to derive the required approximability results.

3.1 Low Degree Advantage

A widely used general condition for proving approximability is due to Hast [8]. A simplified proof was also given by Austrin et. al.[14] using an SDP rounding algorithm. This condition was also used in the study of approximability of symmetric predicates by Guruswami and Lee [6].

▶ **Theorem 10** ([8, 14]). *Let $f : \{-1,1\} \to \{0,1\}$ be a predicate. Suppose there exists $\eta \in \mathbb{R}$, such that*

$$\frac{2\eta}{\sqrt{2\pi}}\cdot\sum_i \hat{f}(\{i\})\cdot x_i + \frac{2}{\pi}\cdot\sum_{i<j}\hat{f}(\{i,j\})\cdot x_i x_j \;>\; 0$$

for all $x \in f^{-1}(1)$. Then f is approximable.

We show that the weak approximability analogue of the above theorem follows directly from Lemma 9.

▶ **Theorem 11.** *Let $f : \{-1,1\} \to \{0,1\}$ be a predicate. Suppose there exists $\eta \in \mathbb{R}$, such that*

$$\frac{2\eta}{\sqrt{2\pi}}\cdot\sum_i \hat{f}(\{i\})\cdot x_i + \frac{2}{\pi}\cdot\sum_{i<j}\hat{f}(\{i,j\})\cdot x_i x_j \;>\; 0$$

for all $x \in f^{-1}(1)$. Then f is weakly approximable.

Proof. Suppose f is strongly approximation resistant. Then, by Theorem 8, there exists a vanishing measure Λ on $\mathbb{C}(f)$.

We first apply Lemma 9 with $t = 1$. Note that this case corresponds to $|S| = 1$. The matrices ζ_S are 2×2 matrices with diagonal entries 1 and off-diagonal entries equal to $\zeta(0, i)$ when $S = \{i\}$. We take the function $h(M) = M(0, 1)$ (equal to the off diagonal entry). Since there are no nontrivial permutations, and multiplying row 1 and column 1 by $b \in \{-1, 1\}$ multiplies $M(0, 1)$ by b, h is an odd symmetric function. Thus, we get

$$\mathop{\mathbb{E}}_{\zeta \sim \Lambda}\left[\sum_{i \in [k]} \hat{f}(\{i\}) \cdot h(\zeta_{\{i\}})\right] \;=\; \mathop{\mathbb{E}}_{\zeta \sim \Lambda}\left[\sum_{i \in [k]} \hat{f}(\{i\}) \cdot \zeta(0, i)\right] \;=\; 0\,.$$

Similarly, for the case of $t = 2$, we consider the function $h(M) = M(1, 2)$. The only nontrivial permutation of $1, 2$ swaps the two indices. Thus, for symmetric matrices ζ_S with $|S| = 2$, this is an odd symmetric function. Hence, we get

$$\mathop{\mathbb{E}}_{\zeta \sim \Lambda}\left[\sum_{i<j} \hat{f}(\{i, j\}) \cdot h(\zeta_{\{i,j\}})\right] \;=\; \mathop{\mathbb{E}}_{\zeta \sim \Lambda}\left[\sum_{i<j} \hat{f}(\{i, j\}) \cdot \zeta(i, j)\right] \;=\; 0\,.$$

Combining the two conditions, we get

$$\mathop{\mathbb{E}}_{\zeta \sim \Lambda}\left[\frac{2\eta}{\sqrt{2\pi}} \cdot \sum_{i} \hat{f}(\{i\}) \cdot \zeta(0, i) + \frac{2}{\pi} \cdot \sum_{i<j} \hat{f}(\{i, j\}) \cdot \zeta(i, j)\right] \;=\; 0\,.$$

Let ζ_0 denote the matrix $\mathbb{E}_{\zeta \sim \Lambda}[\zeta]$. Then, by linearity of expectation

$$\frac{2\eta}{\sqrt{2\pi}} \cdot \sum_{i} \hat{f}(\{i\}) \cdot \zeta_0(0, i) + \frac{2}{\pi} \cdot \sum_{i<j} \hat{f}(\{i, j\}) \cdot \zeta_0(i, j) \;=\; 0\,.$$

Since $\mathbb{C}(f)$ is a convex polytope, $\zeta_0 \in \mathbb{C}(f)$. Thus, there exists a distribution μ_0 with $\mathit{supp}(\mu_0) \subseteq f^{-1}(1)$ satisfying $\zeta_0(0, i) = \mathbb{E}_{x \sim \mu_0}[x_i]$ and $\zeta_0(i, j) = \mathbb{E}_{x \sim \mu_0}[x_i \cdot x_j]$ for all $i, j \in [k]$. Thus, the above condition can we written as

$$\mathop{\mathbb{E}}_{x \sim \mu_0}\left[\frac{2\eta}{\sqrt{2\pi}} \cdot \sum_{i} \hat{f}(\{i\}) \cdot x_i + \frac{2}{\pi} \cdot \sum_{i<j} \hat{f}(\{i, j\}) \cdot x_i x_j\right] \;=\; 0\,,$$

which is a contradiction since the inner quantity is positive for all $x \in f^{-1}(1)$ by assumption. ◀

3.2 Symmetric Predicates

Recall that $f : \{-1, 1\}^k \to \{0, 1\}$ is a symmetric predicate if permuting the input bits of x does not change the value of $f(x)$. Alternatively, $f(x)$ only depends on $\sum_i x_i$. We will also use the fact that for a symmetric function f, $\hat{f}(S)$ only depends on $|S|$.

The approximability of symmetric predicates was studied by Guruswami and Lee [6]. They consider both the cases with and without negation. For the case with negation, as considered in this paper, they show that when f is *even* or corresponds to an *interval* (i.e.there is an interval $I \subseteq [-k, k]$ such that $f(x) = 1 \Leftrightarrow \sum_i x_i \in I$), f is approximation resistant if and only if there exists a balanced pairwise independent distribution distribution μ supported in $f^{-1}(1)$. Note that this condition was shown to be sufficient by Austrin and

Mossel [1]. They show that for the cases of intervals and even predicates, this condition is also necessary.

We study a different class of symmetric predicates, which either have non-zero mass on both of the first two Fourier levels i.e.$\hat{f}(\{1\}) \neq 0$ and $\hat{f}(\{1,2\}) \neq 0$, or have $\hat{f}(\{1\}) = \hat{f}(\{1,2\}) = 0$. We show that any such predicate f is approximation resistant if and only if $f^{-1}(1)$ supports a balanced pairwise independent distribution. We first consider the case when $\hat{f}(\{1\}) = \hat{f}(\{1,2\}) = 0$. In this case, it is easy to see that $f^{-1}(1)$ supports a balanced pairwise independent distribution, and hence f is approximation resistant.

▶ **Theorem 12.** *Let $f : \{-1,1\}^k \to \{0,1\}$ be a symmetric predicate such that $\hat{f}(\{1\}) = \hat{f}(\{1,2\}) = 0$. Then, the uniform distribution on $f^{-1}(1)$ is balanced and pairwise independent.*

Proof. Let μ denote the uniform distribution on $f^{-1}(1)$. Then, for any $i \in [k]$

$$\mathop{\mathbb{E}}_{x\sim\mu}[x_i] = \frac{2^k}{|f^{-1}(1)|} \cdot \mathop{\mathbb{E}}_{x\in\{-1,1\}^k}[f(x)\cdot x_i] = \frac{2^k}{|f^{-1}(1)|} \cdot \hat{f}(\{i\}) = 0\,.$$

Similarly, we also have that $\mathbb{E}_{x\sim\mu}[x_i x_j] = 0$ for all $i \neq j$. ◀

Next, we consider the case when both $\hat{f}(\{1\})$ and $\hat{f}(\{1,2\})$ are nonzero.

▶ **Theorem 13.** *Let $f : \{-1,1\}^k \to \{0,1\}$ be a symmetric predicate such that $\hat{f}(\{1\}) \neq 0$ and $\hat{f}(\{1,2\}) \neq 0$. Then f is strongly approximation resistant if and only if $f^{-1}(1)$ supports a balanced pairwise independent distribution.*

Proof. We only need to prove that strong approximation resistance implies the existence of a balanced pairwise distribution supported in $f^{-1}(1)$, since the other direction follows from the result of Austrin and Mossel [1].

Let f be approximation resistant and let Λ be the corresponding vanishing measure on $\mathbb{C}(f)$. For a permutation $\pi : [k] \to [k]$, recall that Λ_π denotes the measure

$$\Lambda_\pi(\zeta) = \Lambda(\zeta_\pi)\,.$$

By the symmetry of the variables in f, if Λ is a vanishing measure, then so is Λ_π. Since the conditions in Theorem 8 are linear in the measure Λ, we get that $\mathbb{E}_{\pi:[k]\to[k]}[\Lambda_\pi]$ is also a vanishing measure. Thus, we can assume without loss of generality that for the given vanishing measure, we have

$$\mathop{\mathbb{E}}_{\zeta\sim\Lambda}[\zeta(0,i)] = \mathop{\mathbb{E}}_{\zeta\sim\Lambda}[\zeta(0,j)] \;\; \forall i \neq j \quad \text{and} \quad \mathop{\mathbb{E}}_{\zeta\sim\Lambda}[\zeta(i_1,j_1)] = \mathop{\mathbb{E}}_{\zeta\sim\Lambda}[\zeta(i_2,j_2)] \;\; \forall i_1 \neq j_1, i_2 \neq j_2\,. \tag{1}$$

As in Theorem 11, we apply Lemma 9 with $t = 1$ using $h(M) = M(0,1)$, and with $t = 2$ using $h(M) = M(1,2)$, to get the conditions

$$\mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[\sum_{i\in[k]} \hat{f}(\{i\})\cdot\zeta(0,i)\right] = 0 \quad \text{and} \quad \mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[\sum_{i<j} \hat{f}(\{i,j\})\cdot\zeta(i,j)\right] = 0\,.$$

Using the symmetry of the Fourier coefficients, and Eq. (1), this gives

$$\hat{f}(\{1\})\cdot \mathop{\mathbb{E}}_{\zeta\sim\Lambda}[\zeta(0,i)] = 0 \quad \text{and} \quad \hat{f}(\{1,2\})\cdot \mathop{\mathbb{E}}_{\zeta\sim\Lambda}[\zeta(i,j)] = 0 \qquad \forall i,j \in [k],\; i \neq j\,.$$

Since $\hat{f}(\{1\}) \neq 0$ and $\hat{f}(\{1,2\}) \neq 0$, we get that $\mathbb{E}_{\zeta \sim \Lambda}[\zeta(0,i)] = 0$ and $\mathbb{E}_{\zeta \sim \Lambda}[\zeta(i,j)] = 0$ for all $i \neq j \in [k]$. Let $\zeta_0 = \mathbb{E}_{\zeta \sim \Lambda}[\zeta]$. As before, we know that $\zeta_0 \in \mathbb{C}(f)$ by convexity and hence there exists μ_0 supported in $f^{-1}(1)$ such that ζ_0 corresponds to the moments of μ_0. Hence,

$$\mathop{\mathbb{E}}_{x \sim \mu_0}[x_i] \;=\; \zeta_0(0,i) \;=\; 0 \quad \text{and} \quad \mathop{\mathbb{E}}_{x \sim \mu_0}[x_i \cdot x_j] \;=\; \zeta_0(i,j) \;=\; 0 \qquad \forall i,j \in [k],\ i \neq j\,.$$

Thus, μ_0 is a balanced pairwise independent distribution supported in $f^{-1}(1)$. ◀

3.3 Monarchy

Next, we consider the Monarchy predicate, which was proved to be approximable by Austrin et. al.[14]. The predicate is a halfspace defined as

$$f(x) \;:=\; \frac{1 + \operatorname{sgn}\left((k-2) \cdots x_1 + x_2 + \cdots + x_k\right)}{2}\,.$$

The predicate is determined by the value of x_1 unless $x_2 = \cdot = x_k = -x_1$. Austrin et. al.considered this predicate as an example of a predicate to which Hast's condition (discussed in the previous section) does not apply. Moreover, it did not seem amenable to the rounding scheme used in the proof of Hast's result and they provide a new rounding algorithm to prove the approximability of this predicate.

We show that the approximability of Monarchy follows from Lemma 9. Moreover, since it is an odd predicate, weak approximability is equivalent to approximability. We shall use the following observation by Austrin et. al.

▶ **Lemma 14** ([14]). *Let f be the monarchy predicate and let μ be a distribution on $\{-1,1\}^k$ with $supp(\mu) \subseteq f^{-1}(1)$. Then for all $i > 1$,*

$$\mathop{\mathbb{E}}_{x \sim \mu}[x_i] \;\geq\; -\mathop{\mathbb{E}}_{x \sim \mu}[x_1]\,.$$

Proof. If x is a satisfying assignment then either $x_1 = 1$, or for all $i > 2$ $x_i = 1$. In both cases, we have $x_i \geq -x_1$ for all $i > 2$. The claim follows by linearity of expectation. ◀

We will also need the following facts about the Fourier coefficients of the Monarchy predicate.

▶ **Lemma 15.** *Let f be the Monarchy predicate as defined above. Then*
1. $\hat{f}(\{1\}) = 1/2 - 1/2^{k-1}$ *and* $\hat{f}(\{2\}) = \cdots = \hat{f}(\{k\}) = 1/2^{k-1}$.
2. $\hat{f}(S) = 0$ *for all S such that* $|S| = 2$.
3. *For S with $|S| = 3$, $\hat{f}(S) = -1/2^{k-1}$ if $1 \in S$ and $\hat{f}(S) = 1/2^{k-1}$ otherwise.*

We can now prove that Monarchy is approximable.

▶ **Theorem 16.** *Let f be the Monarchy predicate as defined above. Then f is approximable using the basic SDP.*

Proof. Suppose that f is not approximable. Then, by Theorem 8, there exists a vanishing measure Λ on the polytope $\mathbb{C}(f)$. For $s \in \{-1,0,1\}$, define the probabilities $p(s) := \Lambda\left(\{\zeta \mid \operatorname{sgn}(\zeta(0,1)) = s\}\right)$. We first prove the following.

▶ **Lemma 17.** *There exist $\beta_1 \geq 1$ and $\beta_0 \geq 0$ such that $p(-1) = \beta_1 \cdot p(1) + \beta_0 \cdot p(0)$. Moreover, we must have $p(-1) > 0$.*

Proof. We apply Lemma 9 for $t = 1$ and the function $h(M) = \operatorname{sgn}(M_{0,1})$. Then, since h is an odd symmetric function, we get

$$\begin{aligned} \mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[\hat{f}(\{1\})\cdot\operatorname{sgn}(\zeta(0,1)) + \sum_{i>1}\hat{f}(\{i\})\cdot\operatorname{sgn}(\zeta(0,i))\right] &= 0 \\ \Rightarrow \sum_{s\in\{-1,0,1\}} p(s)\cdot\mathbb{E}\left[\hat{f}(\{1\})\cdot\operatorname{sgn}(\zeta(0,1)) + \sum_{i>1}\hat{f}(\{i\})\cdot\operatorname{sgn}(\zeta(0,i)) \mid \operatorname{sgn}(\zeta(0,1)) = s\right] &= 0\,. \end{aligned}$$

Using the facts that $\hat{f}(\{2\}) = \cdots = \hat{f}(\{k\})$ and $\zeta_i \geq -\zeta_1\ \forall i > 2$ by Lemma 14, we get

$$p(-1)\cdot\Big(-\hat{f}(\{1\}) + (k-1)\cdot\hat{f}(\{2\})\Big) + p(0)\cdot\Big(a\cdot\hat{f}(\{2\})\Big) + p(1)\cdot\Big(\hat{f}(\{1\}) + b\cdot\hat{f}(\{2\})\Big) = 0\,,$$

for some $a \in [0, k-1]$ and $b \in [-(k-1), k-1]$. Thus, we get $p(-1) = \beta_1\cdot p(1) + \beta_0\cdot p(0)$, where

$$\beta_1 = \frac{\hat{f}(\{1\}) + b\cdot\hat{f}(\{2\})}{\hat{f}(\{1\}) - (k-1)\cdot\hat{f}(\{2\})} \geq 1 \quad \textit{and} \quad \beta_0 = \frac{a\cdot\hat{f}(\{2\})}{\hat{f}(\{1\}) - (k-1)\cdot\hat{f}(\{2\})} \geq 0$$

To prove the second part of the claim, we again apply Lemma 9 with $t = 1$ and $h(M) = M(0,1)$. This gives,

$$\mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[\hat{f}(\{1\})\cdot\zeta(0,1) + \sum_{i>1}\hat{f}(\{i\})\cdot\zeta(0,i)\right] = 0.$$

By the definition of the Monarchy predicate, we also know that for any $\zeta \in \mathbb{C}(f)$,

$$(k-2)\cdot\zeta(0,1) + \sum_{i>1}\zeta(0,i) > 0\,.$$

Using the fact that $\hat{f}(\{i\}) = \hat{f}(\{2\})$ for all $i > 1$, we get

$$\left((k-2) - \frac{\hat{f}(\{1\})}{\hat{f}(\{2\})}\right)\cdot\mathop{\mathbb{E}}_{\zeta\sim\Lambda}[\zeta(0,1)] > 0 \quad\Rightarrow\quad \mathop{\mathbb{E}}_{\zeta\sim\Lambda}[\zeta(0,1)] < 0\,.$$

Hence, we must have $p(-1) = \mathbb{P}[\zeta(0,1) < 0] > 0$. ◀

Next, we apply Lemma 9 with $t = 3$ and $h(M) = \prod_{j=1}^{3} M(0,j)$. This gives

$$\begin{aligned} \mathop{\mathbb{E}}_{\zeta\sim\Lambda}\left[\sum_{|S|=3}\hat{f}(S)\cdot\prod_{i\in S}\operatorname{sgn}(\zeta(0,i))\right] &= 0 \\ \Rightarrow \sum_{s\in\{-1,0,1\}} p(s)\cdot\mathbb{E}\left[\sum_{|S|=3}\hat{f}(S)\cdot\prod_{i\in S}\operatorname{sgn}(\zeta(0,i)) \;\middle|\; \operatorname{sgn}(\zeta(0,1)) = s\right] &= 0\,. \end{aligned} \tag{2}$$

We analyze the terms for each $s \in \{-1, 0, 1\}$ separately. For $s = -1$, we have $\zeta(0,1) < 0$ and hence, $\zeta(0,i) > 0$ for all $i > 1$, by Lemma 14. Since the Fourier coefficients are negative when $1 \in S$ and positive otherwise (Lemma 15), we get that

$$E(-1) = \mathbb{E}\left[\sum_{|S|=3}\hat{f}(S)\cdot\prod_{i\in S}\operatorname{sgn}(\zeta(0,i)) \;\middle|\; \operatorname{sgn}(\zeta(0,1)) = -1\right] = \sum_{|S|=3}\left|\hat{f}(S)\right|\,.$$

For $s = 0$, we have $\zeta(0, 1) = 0$ and hence $\zeta(0, i) \geq 0$ for all $i > 1$. This gives

$$\begin{aligned} E(0) &= \mathbb{E}\left[\sum_{|S|=3} \hat{f}(S) \cdot \prod_{i \in S} \operatorname{sgn}(\zeta(0,i)) \;\middle|\; \operatorname{sgn}(\zeta(0,1)) = 0\right] \\ &= \mathbb{E}\left[\sum_{\substack{|S|=3 \\ 1 \notin S}} \hat{f}(S) \cdot \prod_{i \in S} \operatorname{sgn}(\zeta(0,i)) \;\middle|\; \operatorname{sgn}(\zeta(0,1)) = 0\right] \;\geq\; 0\,, \end{aligned}$$

since $\hat{f}(S) \geq 0$ for all S with $|S| = 3$ and $1 \notin S$. Finally, for $s = 1$, we note that since $\hat{f}(S) < 0$ for $1 \in S$, we must have

$$\left|\sum_{\substack{|S|=3 \\ 1 \in S}} \hat{f}(S) \prod_{i \in S} \operatorname{sgn}(\zeta(0,i))\right| \;<\; \sum_{\substack{|S|=3 \\ 1 \in S}} \left|\hat{f}(S)\right|\,,$$

since $\operatorname{sgn}(\zeta(0,i)) \cdot \operatorname{sgn}(\zeta(0,j))$ cannot be simultaneously negative for all $i, j > 1$. This gives,

$$|E(1)| \;=\; \left|\mathbb{E}\left[\sum_{|S|=3} \hat{f}(S) \cdot \prod_{i \in S} \operatorname{sgn}(\zeta(0,i)) \;\middle|\; \operatorname{sgn}(\zeta(0,1)) = 1\right]\right| \;<\; \sum_{|S|=3} \left|\hat{f}(S)\right| \;=\; E(-1)\,.$$

We will show that this implies a contradiction to Eq. (2). By Lemma 17, we have that $p(-1) = \beta_1 \cdot p(1) + \beta_0 \cdot p(0)$ for $\beta_1 \geq 1$ and $\beta_0 \geq 0$. Thus, we have

$$\begin{aligned} & p(-1) \cdot E(1) + p(0) \cdot E(0) + p(1) \cdot E(1) \\ &= (\beta_1 p(1) + \beta_0 p(0)) \cdot E(-1) + p(0) \cdot E(0) + p(1) \cdot E(1) \\ &\geq p(1) \cdot (\beta_1 E(-1) - |E(1)|) + p(0) \cdot (\beta_0 E(-1) + E(0))\,, \end{aligned}$$

which is strictly greater than 0 (thus contradicting Eq. (2)) unless $p(1) = 0$ and $\beta_0 = 0$. However, this would imply that $p(-1) = \beta_1 \cdot p(1) + \beta_0 \cdot p(0) = 0$, which is impossible by Lemma 17. ◀

References

1 Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. In *Proceedings of the 23rd IEEE Conference on Computational Complexity*, pages 249–258, Los Alamitos, CA, USA, 2008. IEEE Computer Society. URL: `http://front.math.ucdavis.edu/0802.2300`, `doi:10.1109/CCC.2008.20`.

2 Siu On Chan. Approximation resistance from pairwise independent subgroups. In *Proceedings of the 45th ACM Symposium on Theory of Computing*, pages 447–456, 2013. `doi:10.1145/2488608.2488665`.

3 Mahdi Cheraghchi, Johan Håstad, Marcus Isaksson, and Ola Svensson. Approximating linear threshold predicates. *ACM Transactions on Computation Theory (TOCT)*, 4(1):2, 2012.

4 Lars Engebretsen, Jonas Holmerin, and Alexander Russell. Inapproximability Results for Equations over Finite Groups. *Theor. Comput. Sci.*, 312(1):17–45, 2004. `doi:10.1016/S0304-3975(03)00401-8`.

5 M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995. Preliminary version in *Proc. of STOC'94*.

6 Venkatesan Guruswami and Euiwoong Lee. Towards a characterization of approximation resistance for symmetric CSPs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, pages 305–322, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.305.

7 Venkatesan Guruswami, Daniel Lewin, Madhu Sudan, and Luca Trevisan. A tight characterization of NP with 3 query PCPs. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 8–17, 1998. doi:10.1109/SFCS.1998.743424.

8 Gustav Hast. *Beating a Random Assignment*. PhD thesis, Royal Institute of Technology, Sweden, 2005.

9 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

10 Johan Håstad. Every 2-CSP Allows Nontrivial Approximation. *Computational Complexity*, 17(4):549–566, 2008.

11 Subhash Khot. Hardness Results for Coloring 3-Colorable 3-Uniform Hypergraphs. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 23–32, 2002. doi:10.1109/SFCS.2002.1181879.

12 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 767–775, 2002.

13 Subhash Khot, Madhur Tulsiani, and Pratik Worah. A characterization of strong approximation resistance. In *Proceedings of the 46th ACM Symposium on Theory of Computing*, pages 634–643. ACM, 2014.

14 Avner Magen, Siavosh Benabbas, and Per Austrin. On quadratic threshold CSPs. *Discrete Mathematics & Theoretical Computer Science*, 14, 2012.

15 Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the 40th ACM Symposium on Theory of Computing*, pages 245–254, 2008.

16 Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 191–199, 2000.

17 Alex Samorodnitsky and Luca Trevisan. Gowers uniformity, influence of variables, and PCPs. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 11–20, 2006.

18 Johan Håstad. On the Efficient Approximability of Constraint Satisfaction Problems. In *Surveys in Combinatorics*, volume 346, pages 201–222. Cambridge University Press, 2007.

19 Uri Zwick. Approximation Algorithms for Constraint Satisfaction Problems Involving at Most Three Variables per Constraint. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 201–210, 1998.

Every Property of Outerplanar Graphs is Testable

Jasine Babu[1], Areej Khoury*[2], and Ilan Newman[3]

1 **Department of Computer Science and Engg, Indian Institute of Technology Palakkad, India**
`jasine@iitpkd.ac.in`
2 **Department of Computer Science, University of Haifa, Haifa, Israel**
`areejkhoury@csweb.haifa.ac.il`
3 **Department of Computer Science, University of Haifa, Haifa, Israel**
`ilan@cs.haifa.ac.il`

Abstract

A D-disc around a vertex v of a graph $G = (V, E)$ is the subgraph induced by all vertices of distance at most D from v. We show that the structure of an outerplanar graph on n vertices is determined, up to modification (insertion or deletion) of at most ϵn edges, by a set of D-discs around the vertices, for $D = D(\epsilon)$ that is independent of the size of the graph. Such a result was already known for planar graphs (and any hyperfinite graph class), in the limited case of *bounded degree* graphs (that is, their maximum degree is bounded by some fixed constant, independent of $|V|$). We prove this result with no assumption on the degree of the graph.

A pure combinatorial consequence of this result is that two outerplanar graphs that share the same local views are close to be isomorphic.

We also obtain the following property testing results in the sparse graph model:

- graph isomorphism is testable for outerplanar graphs by $poly(\log n)$ queries.
- every graph property is testable for outerplanar graphs by $poly(\log n)$ queries.

We note that we can replace outerplanar graphs by a slightly more general family of k-edge-outerplanar graphs. The only previous general testing results, as above, where known for forests (Kusumoto and Yoshida), and for some power-law graphs that are extremely close to be bounded degree hyperfinite (by Ito).

1998 ACM Subject Classification G.3 Probability and Statistics

Keywords and phrases Property testing, Isomorphism, Outerplanar graphs

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.21

1 Introduction

We study property testing and the related learning problem for some classes of sparse graphs. The theory of property testing in the dense graph model is quite well understood (see [1] and bibliography therein). The theory of sparse graphs is less understood, and, in particular, there is no characterization of what properties can be tested, even for the bounded degree model.

Our starting point is the result in Newman-Sohler [9] stating roughly that every graph property can be tested by constantly many queries for *bounded* degree planar[1] graphs. The result follows a long line of previous results, and uses heavily a basic idea of Onak [10], and

* Research supported by ERC grant 279559.
[1] The result in[9] is for the larger family of hyperfinite graphs containing planar graphs.

Hassidim et al. [4], (a.k.a. “local partition oracle”) showing that a bounded degree graph G can be approximated, up to the deletion of ϵn edges, by a graph G' whose components are all of constant size. Moreover, the graph G' (or a short description of it), can be obtained by making a constant number of queries to the original graph G.

This result is essentially equivalent to two other formulations: the first is that every bounded degree planar graph can be *learned* up to the deletion of ϵn edges, by making a constant number of queries to it. For the other formulation, let D be a constant natural number. The D-local views of a graph G on n vertices is the collection (multiset) of the n unlabelled discs (balls) of radius D around the n vertices. The other, purely combinatorial result, states that for any ϵ, there is a constant $D = D(\epsilon, d)$, such that if two n-vertices d-bounded degree planar graphs G, H, have the same[2] D-local neighbourhoods, then changing at most ϵn edges in G makes it isomorphic to H (we will say that G is ϵ-close to H in this case).

The results above restrict the graphs they are applicable to, in two conceptually different ways. The first is being planar (or hyperfinite). Indeed it is known that this is essential; namely, we know that similar statements as above are wrong for e.g., bounded degree, but otherwise general graphs. The other restriction is being *bounded degree*. The results above (specifically the distance measure) are defined so to be used for sparse graphs - namely of bounded (constant) *average* degree. Bounding the maximum degree is essential for the proof machinery in the papers above, but does not seem to be essential for the results. However, as of now, the only general results for non-bounded degree families of sparse graphs are known only for the much simpler family of Forests [6], and the special power-law graphs of Ito [5] (that are very close to be hyperfinite). In particular, the following, purely combinatorial question proposed by Sohler [12] is still wide open: Suppose that two n-vertex planar graphs H, G have identical D-local views (for some large enough constant D), is it true that the graphs are ϵ-close to be isomorphic? (ϵ-close means that we can change at most ϵn edges in one to make it isomorphic to the other).

We answer this question positively for a subclass of planar graphs that includes forests and outerplanar graphs (and k-edge-outerplanar graphs - to be defined later). We follow coarsely the route used by Newman-Sohler [9], and the generalization of it to non-bounded degree forests by Kusumoto and Yoshida [6]. As an outcome, we also obtain three other results as well: (a) every graph property is testable for this subclass. (b) isomorphism is testable for any two graphs of this subclass. (c) every such graph G can be “learned”, namely one can infer a graph H that is ϵ-close to G. All results using $poly(\log n)$ many queries.

The presentation is arranged as follows: In Section 2 we present the essential definitions, and the tools we use. We then state the formal results in Section 3, along with a road map to the structure of the proof.

2 Notations and Tools

In this paper we consider labelled undirected graphs without multiple edges and self-loops. We use $G = (V, E)$ to denote a graph with vertex set V and edge set E. We will assume by default that $V(G) = [n]$, unless otherwise stated. We will say that a graph is d-bounded degree if its maximum degree is at most d. For a set $S \subseteq V$ we denote by

[2] The result are asserted even when the the D-local neighbourhoods are not the same, but just close enough.

$E(S) = \{(u, v) \in E(G)|\ u \in S,\ v \notin S\}$, and $e(S) = |E(S)|$. A block in a graph G is a maximal 2-connected subgraph of G.

The subclass of planar graphs that is discussed in this paper is that of k-edge-outerplanar graph for some fixed constant k.

▶ **Definition 2.1** (k-Edge-Outerplanar)**.** A graph G is 1-edge-outerplanar if it has a planar embedding in which all vertices of G are on the outer face.

We say that G is k-edge-outerplanar if G has a planar embedding such that if all edges on the exterior face are deleted, the connected components of the remaining graph are all $(k-1)$-edge-outerplanar.

Note: Being 1-outerplanar coincides with the standard definition of being *outerplanar*. However, for $k > 1$, being k-edge-outerplanar is a weaker notion than the standard notion of being k-outerplanar - namely, graphs that have a planar embedding such that the removal of the *vertices* on the outer face results in a $(k-1)$-outerplanar graph. In particular, a graph may be 2-outerplanar, but not k-edge-outerplanar for any given constant k.

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are said to be *isomorphic*, if there is a bijective mapping $\Phi : V_1 \to V_2$ such that $(u, v) \in E_1$, if and only if $(\Phi(u), \Phi(v)) \in E_2$. A graph property is a (possibly infinite) collection of graphs, which is closed under isomorphism. We will consider graph properties of graphs with fixed number of vertices (n in what follows), where the number is growing to infinity. The graphs that are discussed in this paper are all sparse graphs, specifically, they are planar, and hence their average degree is at most 6.

2.1 Property Testing

▶ **Definition 2.2** (Graph distance)**.** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be planar graphs on n vertices. The distance $dist(G_1, G_2)$ is the number of edges that needs to be deleted and/or inserted from G_1 in order to make it isomorphic to G_2.

We extend the definition of $dist(G_1, G_2)$ for the case where G_1 and G_2 have different number of vertices, by adding a sufficient number of isolated vertices to the graph with the lesser number of vertices.

We say that G_1, G_2 are ϵ-far from being isomorphic (or G_1 is ϵ-far from G_2), if $dist(G_1, G_2) > \epsilon n$, where $n = \min\{|V_1|, |V_2|\}$. Otherwise, we say that they are ϵ-close (to being isomorphic).

▶ **Definition 2.3** (ϵ-far)**.** Let Π be any (non-empty) graph property. A graph $G = (V, E)$ is said to be *ϵ-far* from Π, if it is ϵ-far from every $G' \in \Pi$. If G is not ϵ-far from Π, it is said to be *ϵ-close* to Π.

For algorithms in the model that we will discuss in this paper, the input graph, $G = (V, E)$, is given but not known to the algorithm. The vertex set $V = [n]$ is known. The neighbours of each vertex $v \in [n]$ are assumed to be ordered, namely by a list $u_1, \ldots, u_{d(v)}$, where $d(v) = deg(v)$ is the degree of v. The access of the algorithm to the input graph is via 'neighbourhood' queries: A query is to specify a vertex name $v \in [n]$, and $i \in [n]$, on which the answer to the query is the name of the i-th neighbour of v, or a special indication if $deg(v) < i$. We further augment this standard model with an additional type of queries: On a queried vertex v, one gets $deg(v)$. It is easy to see that $deg(v)$ can be determined using the standard model at the cost of $O(\log n)$ queries[3].

[3] A good enough approximation at a better cost would suffice for all our purposes; but we do not use this here, as we do not expect to optimize the query complexity to better than $poly(\log n)$.

The notion of property testing was introduced by Rubinfeld and Sudan [11] and then formally defined by Goldreich, Goldwasser and Ron [3]. A property testing algorithm for property Π, for the model of *sparse* graphs, or *bounded degree* graph model is a (randomized) algorithm that, given query access to a graph G as described above, accepts every graph from Π with probability at least $2/3$, and rejects every graph that is ϵ-far from Π with probability at least $2/3$. If the graph neither has property Π nor is ϵ-far from Π, then a property tester may accept or reject.

2.2 Partitions and the local views of the graph

For a graph $G = (V, E)$, and a set of vertices $S \subseteq V(G)$, $G[S]$ denotes the subgraph induced by S. A partition of a set V is a set of pairwise disjoint non-empty subsets of V whose union is V. For a partition $P = (C_1, C_2, ..., C_r)$ of $V(G)$ we denote by $G[P]$ the graph that is the union of $G[C_i]$. Note that $G[P]$ is disconnected if $r \geq 2$ and is obtained from G by deleting all edges whose endpoints are in different partition classes of P.

Every d-bounded degree planar graph admits a partitioning into small (constant size) connected components by removing a fraction of the edges, by using recursively the Lipton-Tarjan separator [8]. To be useful for property testing and sub-linear approximation algorithms, it would be nice if the features of such partitions could be obtained by some local sampling. Hassidim, Kelner, Nguyen, and Onak in a seminal work, [4], following an earlier work of Benjamini, Schramm and Shapira[2], showed how to construct an oracle to such a partition, that takes a vertex as input and returns in *constant time* the partition class that vertex belongs to.

We will use extensively the local-partition-oracle for d-bounded degree planar graphs, and the related results which we present in what follows.

A connected graph $G = (V, E)$ with a specially identified vertex v, is called *rooted graph* and we sometimes say that G is rooted at v. A rooted graph $G = (V, E)$ has radius D, if every vertex in V has distance at most D from v. Two rooted graphs G and H are isomorphic, if there is a graph isomorphism between H and G that identifies the roots with each other. For a graph $G = (V, E)$, an integer D and a vertex $v \in V$, let $B_G(v, D)$ be the subgraph rooted at v that is induced by all vertices of G that are at distance less or equal to D from v. $B_G(v, D)$ is a graph of radius at most D with root v, and we call it the D-disc around v. The collection (multiset) of the n unlabelled D-discs of G is called the D-local views of G. Note that for d-bounded degree graphs, the number of possible non-isomorphic D-discs is a constant depending on D and d, and does not depend on n.

▶ **Definition 2.4** (Frequency Vector). For integers $d \geq 1$ and $s \geq 1$, a graph is called (d, s)-graph if it is d-bounded degree and has at most s vertices.

Let $\mathcal{F}(d, s) = F^{(1)}, F^{(2)}, ..., F^{(f(d,s))}$ be the family of all non-isomorphic (d, s) planar graphs and let $f(d, s) = |\mathcal{F}(d, s)|$.

For a graph $G = (V, E)$ and a partition P of V that gives a collection of (d, s) components $G[P]$, the P-frequency vector $Freq(G[P])$ is the $f(d, s)$-dimensional vector whose i-th coordinate is the number of (d, s)-components of G that are isomorphic to $F^{(i)}$. Let the normalized P-frequency vector $freq(G[P])$ be the ℓ_1-unit vector $\frac{1}{m} \cdot Freq(G[P])$, where m is the number of (d, s) components[4] in $G[P]$.

[4] In [9] the frequency vector is for rooted components, hence normalization is by dividing into n - the number of vertices. These two notions are interchangeable in terms of the ability to approximate them.

We will use the following theorem considered and proved first by Onak [10], and by Hassidim, Kelner, Nguyen, and Onak [4] on partitions of bounded degree hyperfinite graphs. For the statement below, we use the better bounds achieved by Levi-Ron [7] and we state it here only for the restricted case of planar graphs.

▶ **Lemma 2.5** (Onak's local-partition oracle, [7]). *Let $\epsilon > 0$, and $d \geq 2$. Then there is an $s = s_{2.5}(\epsilon, d) = O(d^2/\epsilon^2)$ and a randomized algorithm (a.k.a "local partition oracle"), $\mathcal{A}$, such that for every d-bounded degree planar graph $G = (V, E)$, algorithm $\mathcal{A}$ produces an implicit partition P so that $G[P]$ is a collection of (d, s)-components.*

Algorithm $\mathcal{A}$ provides a "neighbourhood oracle" to $G[P]$, namely, for a query to a vertex $v \in V(G)$, the algorithm returns the name of a component of $G[P]$ in which v lies in, by doing $(d/\epsilon)^{O(log(1/\epsilon))}$ queries to the graph G, and more specifically to vertices in $B_G(v, poly(1/\epsilon))$.

The total time complexity of a sequence of q queries to the oracle is $q \log q \cdot (d/\epsilon)^{O(log(1/\epsilon))}$ and with success probability $9/10$, the answers are all consistent with a partition P such that $G[P]$ is ϵ-close to G.

Using the local partition oracle, Newman-Sohler [9] proved that the normalized P-frequency vector of $G[P]$, for a (d, s) partition P of a d-bounded degree hyperfinite graph G can be estimated with an additive error of ϵ in its l_1-norm, simply by querying the $D = D(s)$-neighbourhoods (D-discs) around some constant number of randomly chosen vertices in G.

▶ **Definition 2.6.** Let $f, g \in \mathbb{R}^n$ be two vectors. We say that g λ-approximates f if $|f - g|_1 = \sum_1^n |g_i - f_i| \leq \lambda$.

The following lemma is a restatement of Lemma 5.2 of [9] for the restricted case of planar graphs (originally stated in [9] for hyperfinite graphs). Here we do not specify the function types explicitly, so to make the lemma more readable.

▶ **Lemma 2.7** ([9]). *Let $G = (V, E)$ be a d-bounded degree planar graph, and $\epsilon \in (0, 1)$ any constant. Let $s = 100d^2/\epsilon^2$. Then there are values $D_{2.7} = D_{2.7}(\epsilon, d)$, $q_{2.7} = q_{2.7}(\epsilon, d)$ and a randomized algorithm* Sampler*, that accesses the graph G by querying independently $q_{2.7}$ random vertices of G and exploring the $D_{2.7}$-discs around them. The algorithm outputs a frequency vector $\tilde{f}$ with the following properties.*

With probability at least $4/5$ (over the internal coins of the algorithm) the following two events occur simultaneously: (a) the output vector $\tilde{f}$ ϵ-approximates the normalized (d, s)-frequency vector $freq(G[P])$ of the graph $G[P]$, where P is a partition of G into (d, s)-components. (b) $G[P]$ is ϵ-close to G.

Finally, to close the cycle, the following simple claim shows why an approximation of $freq(G[P])$ as above is useful.

▶ **Claim 2.8** ([9]). *Let $s \geq 1$ be an integer and let $0 < \lambda < 1$. Let G and H be two graphs that are each a union of (d, s)-graphs on n vertices such that their normalized frequency vectors (for the corresponding partitions into components) f, g respectively have $|f - g|_1 \leq \lambda$. Then G and H are λ-close.*

For non-bounded degree outerplanar graphs it is not always possible to delete ϵn edges so that in the resulting graph all components are of constant sizes. E.g., consider the star of n vertices. Hence, we allow some more complex pieces in the partitions. This motivates the following definition that is introduced in [6] for partitions of forests.

▶ **Definition 2.9** ((d,s)-union)**.** A graph $G=(V,E)$ is a (d,s)-rooted graph if G contains a (unique) vertex v with $deg(v) \geq d+1$ and each connected component of $G \setminus \{v\}$ is (d,s) graph. The unique vertex v (with $deg_G(v) \geq d+1$) is called the *root* vertex of G and is denoted by $root(G)$.

A graph is a (d,s)-union if it is a vertex disjoint union of (d,s)-rooted components and (d,s)-components.

▶ **Definition 2.10** (Multiway-Cut)**.** For a graph G and a set of vertices $T \subseteq V(G)$ a T-multiway cut is a set of edges $E' \subseteq E(G)$ such that in the graph $G \setminus E'$ no two vertices from T are in the same connected component.

3 Global Partitions

In this section we prove the main structural theorem stating that every k-edge outerplanar graph is close to a k-edge outerplanar (d,s)-union, for some constants d,s (which depend only on ϵ and k). For clarity we present in this version the statements, and results for outerplanar graphs (rather than k-edge outerplanar graphs). The generalization to k-edge-outerplanar graphs is immediate, but will not be presented here. Note, however that the constants d and s will depend also on k when the generalization is done.

▶ **Theorem 3.1.** *Every outerplanar graph G is ϵ-close to a graph G' that is an outerplanar (d,s)-union for some $d=d(\epsilon)$ and $s=s(\epsilon)$.*

We note that this does not immediately imply that every such G has a 'short' (constant size) description, as each component of G' may have a root of different and unbounded degree. It does not imply also, that such a "close" graph G' can be "learned" from the local views in G. Thus, this is not directly applicable for property testing, but could be of independent interest. We will prove the theorem, and provide positive answer for the two additional properties above, namely that G' can be learned from the local views, and that it has a "short" description.

Before we present the proof of Theorem 3.1 we make some observations about outerplanar graphs which provides the core tool for the proof as well as the motivation for the definition of k-edge outerplanar graphs.

For a graph $G=(V,E)$ and $a,b \in V$ let $c(a,b)$ denote the minimum edge cut in G, separating a and b. The following basic Lemma 3.2 is used, via a chain of reductions, to prove Corollary 3.3 (See appendix for further details and proofs).

▶ **Lemma 3.2.** *Let $G(V,E)$ be 2-connected outerplanar graph, $s,t \in V$ such that (s,t) is an edge of the outer face in the embedding of G as an outerplanar graph. Then $c(s,t) \leq \lfloor(\log(|V|+1))\rfloor$.*

▶ **Corollary 3.3.** *Let $G(V,E)$ be a connected outerplanar graph and $U,W \subsetneq V$ be disjoint subsets of vertices. Suppose that U is an independent set in G Then, there is a U-multiway cut of size at most $2(|U|-1)\log(2|W|+1)$ in the graph $G[W \cup U]$.*

▶ **Claim 3.4.** *Let G be a bipartite outerplanar graph with bipartition A,B. If degree of each vertex in B is at least two, then $|B| \leq 4|A|$ and hence, G has at most $15|A|$ edges.*

We note that reducing the constants 4 and 15 in the above claim can be reduced by a factor of two, but this is of little interest in our context. We prefer the current proof of Claim 3.4 in the appendix, because its generalization to handle k-edge-outerplanar graphs is easy.

Algorithm 1 Given d, ϵ, and an outerplanar graph $G(V, E)$ this algorithm returns an outerplanar (d, s)-union graph $G'(V, E')$, such that G' is obtained from G by removing $f(s, d, \epsilon) \cdot n$ edges, where $s = s_{2.5}(\epsilon/4, d)$.

```
1: procedure GlobalPartition(G)
2:     Let $V^h = \{v \in V \mid deg_G(v) > d\}$, and $V^l = V \setminus V^h$.
3:     Let $E_1 = E(G[V^h])$ be the set of edges with both endpoints in $G[V^h]$, and let $G_1$ be
       the graph obtained from $G$ by deleting $E_1$.
4:     Find a $(\epsilon/4, s)$-partition of $G[V^l]$. Such a partition exists, and, in particular, as
       asserted in Lemma 2.5, a (local) oracle to such partition can be found. Let $G_2$ be the
       graph resulting from $G[V^l]$ after partitioning. $G_2$ is a disjoint union of $(d, s)$-components.
5:     Replace $G[V^l]$ by $G_2$ in $G_1$, that is: let $G_3 = (V, E_2 \cup F)$, where $E_2 = E(G_2)$ and
       $F = E(G) \cap (V^h \times V^l)$. Namely $F$ contains all edges of $G$ with exactly one endpoint in
       $V^h$ and one endpoint in $V^l$.
6:     Finally, obtain $G'$ from $G_3$ by removing for each component $C$ of $G_2$ a minimum size
       $V^h$-multiway cut in the graph $G_3[C \cup (V^h \cap N(C))]$.
7: end procedure
```

Now we are ready to prove Theorem 3.1. The proof will be algorithmic, namely, *Algorithm 1* below will produce the required G' that is close to G.

▶ **Theorem 3.5.** *Let $\epsilon \in (0, 1)$ be any constant. Let $G = (V, E)$ be an outerplanar graph, $d = d(\epsilon) = O(\frac{1}{\epsilon^2})$, $s = s_{2.5}(\epsilon/4, d) = O(d^2/\epsilon^2)$. Then Algorithm 1 produces a (d, s)-union subgraph G' of G, that is ϵ-close to G with probability better than 0.9.*

Proof. Since G is planar, it follows that $|E(G)| \le 3n$. This implies that $|V^h| \le 6n/d$. Since the graph is planar then $G[V^h]$ is planar too. Hence, at most $3|V^h| \le 18n/d$ edges are removed in step 3 of the algorithm. We fix $d = O(\frac{1}{\epsilon^2})$ to be sufficiently high, so to make sure that at most $\epsilon^2 n/10$ edges are removed in step 3.

Applying the global partition in step 4 with parameters $\epsilon_1 = \frac{\epsilon}{4}$ and d we obtain, with success probability 0.9, a graph G_2 that is a a union of (d, s)-components, and that is ϵ_1-close to $G[V^l]$. This defines the graph G_3 in step 5 of the algorithm.

By Claim 3.4, the number of connected components in G_2 with at least two neighbours in V^h in the graph G_3 is at most $4|V^h|$. If each of these components is contracted to a single representative vertex for the component, after removal of parallel edges and self loops, there are only $15|V^h|$ edges between the representative vertices and V^h.

For step 6, observe that if for each component C of $G_3[V^l]$ we get a $N(C) \cap V^h$-multiway cut M_C in $G_3[V(C) \cup (N(C) \cap V^h)]$, then $M = \cup_C M_C$ will be a V^h-multiway cut in G_3. Moreover, we can restrict our attention to only components which have at least two neighbours in V^h. As explained in the paragraph above, the number of such components is only $4|V^h|$ and $\sum_C |N(C) \cap V^h|$ is at most $15|V^h|$. For each such component C, we have $|M_C| \le |N(C) \cap V^h| \cdot (2 \log(2s + 1))$ by Corollary 3.3. From this, it follows that $|M| \le 15|V^h|(2 \log(2s + 1))$. Since $s = O(d^2/\epsilon^2)$, a proper choice of d ensures that $|M| \le \epsilon n/3$.

Thus, the total number of edges removed in all steps of the algorithm is at most ϵn, implying that the resultant graph G' is ϵ-close to G.

After applying the partitioning oracle in step 4 the size of every connected component in G_2 is at most s. Since V^h becomes an independent set after step 3, after executing step 6, no two vertices in V^h have a path between them in G'. Therefore, each component of G' has at most one vertex of degree greater than d. Therefore G' is a (d, s)-union for d and s as above. ◀

We note that in the proof above, step 4 of the algorithm, which is the only random part, may be replaced with any deterministic partition (e.g., recursively removing edges connected to a good enough separator). We used random local partition, in the spirit of Onak [4], looking ahead, to hint to the fact that the partition can actually be done in a *distributed* manner, and hence "approximated" locally. The same is also true with respect to step 6, where a global multiway cut could be taken. It is possible to do step 6 in a distributed way and locally, because a component C of G_2 has at most $d \cdot s$ (which is a constant) neighbours in V_h, and hence $G_3[C \cup (V^h \cap N(C))]$ is a graph of constant size.

4 From global partition to Local partition

Let $G = (V, E)$ be an outerplanar graph. Recall that our goal is two fold: the first is to roughly "learn" G from its *local views*. Learning here means to find a graph G' that is a (d, s)-union and that is close to G, as asserted by Theorem 3.1. Conceptually this implies that two graphs with the same local views are close to be isomorphic (some extra care should be taken here). The 2nd goal is to find the above approximating G' using a small number of queries. Conceptually this immediately implies a property testing mechanism for all properties.

This is summed up in the following theorems.

▶ **Theorem 4.1.** *For every $\epsilon > 0$ there is a $D = D_{4.1}(\epsilon)$, $s = s_{4.1}(\epsilon), d = d_{4.1}(\epsilon)$, $q = q_{4.1}(\epsilon, n) = O(poly(\log n))$, and a randomized algorithm* Approx *that on an outerplanar graph $G = (V, E)$ on n vertices:*

- Approx *outputs an outerplanar (d, s)-union graph G^*.*
- Approx *does random queries to q vertices in G, and only inside the D-disc around the above vertices.*
- *With success probability at least 0.9, G^* will be ϵ-close to G.*

▶ **Theorem 4.2.** *For every $\epsilon > 0$ there is a $D_{4.1} = D(\epsilon)$, $q = q_{4.1}(\epsilon, n) = O(poly(\log n))$, and a randomized algorithm Tester, that on two outerplanar graphs G, H on n vertices, it accepts if H is isomorphic to G and rejects if H is ϵ far from G with error probability at most* $1/3$*.*

The algorithm Tester does q random queries to q vertices in G and H, only inside the D-disc around (some of) the above vertices.

▶ **Theorem 4.3.** *For every $\epsilon > 0$ and a graph property Π, of graphs on n vertices, there is a $D = D_{4.1}(\epsilon)$, $q = q_{4.1}(\epsilon, n) = O(poly(\log n))$, and a randomized algorithm T_Π, that accepts every outerplanar graph G having the property, and rejects every outerplanar graph G that is ϵ-far from Π, with error probability* $1/3$*.*

The algorithm T_Π does q random queries to q vertices in G, and only inside the D-disc around (some of) the above vertices. Moreover, the queries to G are oblivious of Π: only the final decision once the q queries are done, is dependent on Π.

▶ **Theorem 4.4.** *For every $\epsilon > 0$ there is a $D = D_{4.1}(\epsilon)$ such that if two outerplanar graphs G, H on n vertices, have identical D-views then H is ϵ-close to G.*

We note that analogue theorems for planar d-bounded-degree graphs are given in [9]. However, unlike the case for d-bounded-degree planar graphs, that have constant size approximations in form of a union of (d,s)-components, a (d,s)-union graph does not necessarily has a short description. This is due to the fact that the degree of the root of every component may be arbitrary number in $[n-1]$, and hence there are non-constant many types of possible components (let alone their number). To overcome this difficulty we define an ϵ-net for (d,s)-union graphs, namely, a set $\mathcal{G}(d,s)$ of (d,s)-union graphs (of relatively short description), and show that for every G' as above, there is a graph $G'' \in \mathcal{G}(d,s)$ that is close to G'.

Further we will show that G'' can be obtained from the original G by sampling. As will turn out, this sampling can be restricted to randomly sampling a relatively small number of vertices ($poly(\log n)$), in some constant-diameter discs in G. Hence this will provide the "locality" that is stated as desirable above. A similar method in nature, was used by Kusumoto and Yoshida, [6], for unbounded degree forests.

We need the following definitions.

▶ **Definition 4.5.** [γ-layered (d,s) union graphs] Let $\gamma > 1$ be a constant. A γ-layered (d,s) union graph is a (d,s) union graph in which all high-degree vertices have degrees that are γ-powers, namely, in the set $\{\gamma^i\}_{i=\alpha}^{L}$ where $L = \lfloor \log_\gamma n - 1 \rfloor$ and $\alpha = \min\{i | \gamma^i \geq d+1\}$. We denote by G^0 the components of a (d,s) union G that are (d,s) graphs.

In the above definition, all γ^i are assumed to be integral. This is achieved by rounding if necessary. We do not explicitly write this rounding to increase readability. The extra rounding will not affect any of our results.

The role of γ-layered graphs is obvious from the following claim.

▶ **Claim 4.6.** *For every $\epsilon > 0$ there is a $\gamma = \gamma_{4.6}(\epsilon, d) \in (1,2)$ such that every (d,s)-union graph G is ϵ-close to a γ-layered (d,s)-union graph.*

Proof. Let $\gamma \in (1,2)$ to be defined later, and let $G^0, , ..., G^L$ be a partition of G, where G^i, $i = \alpha, \ldots, L$ contains all (d,s)-rooted components C with $deg(root(C)) \in (\gamma^{i-1}, \gamma^i]$ and G^0 contains the (d,s) components. Let n_i be the number of components in G^i.

Consider each G^i separately. For $i \geq \alpha$, and for each component C in G^i we add at most $\gamma^i - \gamma^{i-1}$ isolated vertices and link them to $root(C)$. This obviously makes the graph γ-layered. Thus for G^i we added at most $n_i \cdot (\gamma^i - \gamma^{i-1}) = n_i(\gamma - 1)\gamma^{i-1}$ edges. Note, however, that $n_i \cdot \gamma^{i-1} \leq |V(G^i)|$, as every component in G^i contains a vertex with degree larger than γ^{i-1}.

For G^0 we do not need to change anything. This results in a total number of edge changes bounded by $(\gamma - 1) \cdot \sum_{i \geq \alpha} |V(G^i)| \leq (\gamma - 1)n$. Hence setting $\gamma \leq (1+\epsilon)$ implies the claim. ◀

Now, to define a short description (a.k.a. "sketch") for a γ-layered (d,s) union graph, all we need is to define the structure of G^i, $i = \alpha, \ldots, L$, and that of G^0. For the latter, a good sketch is the (d,s)-frequency vector of G^0, (or a good approximation of it), as being done in [9]. This will also become clear as a special case in what follows. For $G^i, i \geq \alpha$, we only need to define the structure of C for each component $C \in G^i$.

Note that $C \setminus root(C)$ is a union of (d,s) components, each with some marked subset of vertices, indicating the neighbour set of $root(C)$. Since there are constantly many possible (d,s) graphs, there are also constantly many (d,s)-graphs with marked vertices. Hence, each component C of G^i is defined by the (d,s) frequency vectors of the marked components, $\{Freq(C \setminus root(C))\}_{C \in G^i}$. Still, computing for each $C \in G^i$ its frequency vector would be too demanding. Instead we will approximate this vector, using the easy Claim 2.8. Doing this will bring us two advantages; the first is that we will still get a component C' which is close enough to C, but which we will be able to afford (in terms of number of queries). The second and more important feature is the reduction in the number of types of components to a constant, thereby making it possible to approximate G^i by estimating the number of components of each of these constantly many types.

This is summed up in what follows:

Recall that for fixed d and s we set $f(d,s) = |\mathcal{F}(d,s)|$ (which is a constant), where $\mathcal{F}(d,s)$ is the set of all possible outerplanar(d,s)-graphs. We now add a boolean marking of vertices in each (d,s)-graph. This boolean marking will be used later to indicate which vertices in the component are connected to its root in a rooted component (if at all). Hence the histogram, and the frequency vector, is of dimension $2^s \cdot f(d,s)$, since corresponding to each graph in $\mathcal{F}(d,s)$, we have also to specify which subset of vertices in it are marked (have 1-marking).

For fixed $\gamma > 1$ and d,s, let $G = G^0 \cup (\cup_{i=\alpha}^{L} G^i)$ be a γ-layered (d,s) union graph, and fix an $i \in \{\alpha,\dots,L\}$. As explained above, each component $C \in G^i$ is completely defined by its (d,s) frequency vector $Freq(C) \in [n]^{2^s f(d,s)}$, where the marked vertices in each (d,s)-component of $C \setminus root(C)$ are the vertices that are connected to $root(C)$. Let $freq(C) = Freq(C)/(\text{sum of coordinates of } Freq(C))$. Note that $||freq(C)||_1 = 1$.

Let $0 < \delta < 1$ be small enough constant (to be defined later), and $N(\delta)$ be a δ-net for the ℓ_1-unit ball of dimension $2^s f(d,s)$. Obviously such an $N(\delta)$ whose size is a constant that depends only on δ, d, s exists. For example, take $N(\delta) = \{\delta\overline{x} \mid \overline{x}$ is a $(2^s f(d,s))$-dim vector of integral coordinates whose absolute values sum up to $1/\delta\}$.

For $Freq(C)$ as above, we define its *δ-normalized approximation* as a closest vector in $N(\delta)$ to $freq(C)$ (in case of tie choose an arbitrary closest vector). Thus, we have a mapping that maps each component C of G^i to a constant size alphabet (of size $|N(\delta)|$), and hence G^i is mapped into a vector $LFreq(G^i) \in [n]^{|N(\delta)|}$, where the jth coordinate is the number of components C in G^i that have type$=j$ as their δ-normalized approximation. Again, we normalize as follows: Let n_i be the number of components in G^i, we let $lfreq(G^i) = \frac{1}{n_i} \cdot LFreq(G^i)$.

▶ **Claim 4.7.** *Let G^i be the ith layer, $i > 0$, of a γ-layered (d,s)-union graph as above. Let $\epsilon > 0$. Then there is a constant $\nu = \nu_{4.7} = \nu_{4.7}(d,s,\delta,\epsilon) \in (0,1)$ such that if $|\tilde{n_i} - n_i|\gamma^i \leq \nu \cdot \max\{n_i\gamma^i, n/L\}$, and $|\tilde{f} - lfreq(G^i)|_1 \leq \nu$, the graph $\tilde{G^i}$ that is defined as stated below has $dist(G^i, \tilde{G^i}) \leq \epsilon \cdot \max\{n_i\gamma^i, n/L\}$.*

Here $\tilde{G^i}$ is the following graph: let $F = \tilde{n_i} \cdot \tilde{f} = (\tilde{m}_1, \dots, \tilde{m}_{|N(\delta)|})$ and for $j = 1\dots, |N(\delta)|$, let C_j be a rooted component whose frequency vector is the j-type frequency vector. Then for $j = 1\dots, |N(\delta)|$ we include $\lceil \tilde{m}_j \rceil$ disjoint copies of C_j in $\tilde{G^i}$.

Note that the claim only asserts an *additive* error between G^i and $\tilde{G^i}$ that is not necessarily proportional to the size of G^i. However, since there are L "layers", the average G^i has n/L vertices. For G^i larger than the average, the above approximation is with a ν-multiplicative error. For G^i smaller than the average, the additive error is a fraction of the average, which we will be able to afford.

Proof. Let $G^i, \tilde{G^i}$ as above , and let $m_j = LFreq(G^i)_j = f_j \cdot n_i$ be the number of components in G^i of type j. Namely $lfreq(G^i) = (f_1, \dots, f_{|N(\delta)|})$. Let $\Delta = \max\{n_i\gamma^i, n/L\}$. A close isomorphism between $G^i, \tilde{G^i}$ is clear: we map for each type j, the corresponding matching components, leaving $|\tilde{m}_j - m_j|$ components unmatched. For the unmatched components we remove all edges and make the corresponding nodes isolated points. Hence the contribution of type j to the distance (edge-count) is bounded by $|m_j - \tilde{m}_j| \cdot \gamma^i \cdot e(d,s)$ (disregarding here errors due to non-integrality), where $e(d,s)$ is the maximum number of edges in a (d,s) graph (which is constant).

Summing this over all $j \in [|N(\delta)|]$ one gets:

$$\begin{aligned} dist(\tilde{G}^i, G^i) &\leq \sum_{j=1}^{|N(\delta)|} |m_j - \tilde{m_j}| \cdot \gamma^i \cdot e(d,s) \leq e(d,s)\gamma^i \sum_j |\tilde{n}_i \tilde{f}_j - n_i f_j| \\ &\leq e(d,s)\gamma^i \sum_j |\tilde{n}_i \tilde{f}_j - \tilde{n}_i f_j + \tilde{n}_i f_j - n_i f_j| \\ &\leq e(d,s)\gamma^i \tilde{n}_i \cdot \nu + e(d,s)\gamma^i |\tilde{n}_i - n_i| \cdot |lfreq(G^i)|_1 \\ &\leq e(d,s) \cdot 2\nu\Delta + e(d,s)\nu\Delta \end{aligned}$$

Now if we set $\nu \leq \frac{\epsilon}{3e(d,s)}$ we get the asserted claim. ◀

We now restate Theorem 4.1 in a more detailed version, and present its proof.

▶ **Theorem 4.8.** *For every $\epsilon > 0$ there is a $D_{4.1} = D(\epsilon)$, $s = s_{4.1}(\epsilon), d = d_{4.1}(\epsilon)$, $\gamma_{4.1}(\epsilon)$, $q = q_{4.1}(\epsilon, n) = O(poly(\log n))$, there is a randomized algorithm* Approx*, that on an outerplanar graph $G = (V, E)$ on n vertices, outputs an outerplanar γ-layered graph G^*.*

The algorithm Approx *does q random queries to q vertices in G, and only inside the D-disc around (some of) the above vertices.*

It holds that with success probability at least 0.9, G^* will be ϵ-close to G.

Proof of Theorem 4.8 (Sketch). We start by defining G' as the (d,s)-union graph obtained by *Algorithm 1*, for $\epsilon' = \epsilon/10$. We do not know G', but we know how it would have been formed by *Algorithm 1*. We also know that with high probability it would be $\epsilon/10$-close to G. By Claim 4.6, this implicitly defines a γ-layered graph G^* that is close to G', for a suitably small γ. Let $G^* = G^0 \cup \cup_{i=\alpha}^{L} G^i$, and for $i = 0, \alpha, \ldots L$, let n_i be the number of components of G^i, and $f_i = lfreq(G^i)$. Let $roots(G')$ be the set of high-degree vertices in G', namely the roots of the (d,s)-components of G'.

The main part of the algorithm, is algorithm Sampler that is described in the appendix. Algorithm Sampler aims at choosing a vertex y that is distributed uniformly at random among the $roots(G')$ that are in any given layer of G^*. For such y it will also approximate its degree in G' accurately enough, and while doing this it will also approximate $freq(y)$, the approximated frequency vector of y (although this is defined w.r.t G^* rather than G').

Once this is done, approximation $n_i, lfreq(G^i)$ as required by Claim 4.7, for every $i \leq L$ is straight forwards: we just sample q random y's as above, for q large enough ($q = poly(\log n)$) and for each obtain its $freq$ and degree. Then, by normalizing, the proportion of such vertices that are in any interval $[\gamma^{i-1}, \gamma^i)$ is a good approximation of n_i, while the proportion of each type of $freq(y)$ gives an approximation of $lfreq(G^i)$. Finally, having these estimates, Claim 4.7 ends the proof.

The idea behind the Sampler is also simple. We choose a high-degree vertex y at random from V^h by sampling uniformly an edge (v,y) where $v \in V^l$ and $y \in V^h$. Once we have such y, we sample a random neighbour v of it of small degree, discover the component of v, in $G'[V^l]$ by running the local partition oracle for d-bounded graphs, and deleting the multiway cut. As a result, a random (d,s) component connected to y in G' is found (or a conclusion that v is not in the (d,s)-component connected to y). Having found such a random component, we repeat the process for q independent times, which allows us to estimate (again by Chernoff), the degree of y in G', and its frequency $freq(y)$.

Some extra care should be taken since the Sampler cannot succeed for every y that is a root of G'. Consider a vertex y for which $deg_G(y) >> deg_{G'}(y)$. For such y, for most neighbours v of y, their components in the (d,s) partition of $G[V^l]$ (possibly after deleting the edges in the relevant multiway cut) will not be connected to y, and $\deg_{G'}(y)$ might not be estimated correctly. Such vertices we call "bad". In the proof of correctness of the algorithm Approx that outputs G^* we will show that while bad vertices contribute some additional increase in the distance between the estimated G^* and G, this increase in distance is small enough, so that the produced G^* will be (w.h.p) as needed.

We end this very high level description of the sampling process by two notes. The first is that we need to approximate every (large) G^i. Namely, we need to decrease the failure probability in each large G^i to $O(1/\log n)$.

The second remark is that, a similar estimation in spirit (although starting from a forest rather than the more general outerplanar graph), is done in [6], but using a different and finer metric.

For further details, see the algorithm Approx in the appendix. ◀

▶ Remark. It is a suitable point here to note the difference of the results in this paper up to this point, and the results for d-bounded hyperfinite (or planar) graphs of [9, 4]. In the cited papers, a local oracle (in the sense of Onak, as described above) is obtained for the (d, s)-graph H that is close to G. This local oracle is used to approximate the frequency vector of the components in a straight forwards way, by sampling. In our case, a local oracle to the (d, s) union graph (γ-layered) graph G^* is not obtained; instead it is only "nearly" obtained. It fails to produce a local oracle exactly for the bad y's as explained in the proof. Namely, let u be a high-degree vertex in G^* (and hence in G too). It could be that many edges adjacent to u in G are absent from G^*. Hence when asking for a random neighbour of u, the sampler above may not succeed in finding one.

We present now the proofs of Theorems 4.3 and 4.2. These proofs follow from Theorem 4.8 exactly as in [9]. Before getting into the proof, it should be noted that in the standard model, we are concerned only about the number of queries to the input graph and not about the running time of the algorithm. The number of vertices in the input graphs is also an information available.

Proof of Theorem 4.3. Let Π be any graph property, and let Π_n be its restriction to graph on n vertices. An ϵ-tester T_Π for Π_n for outerplanar graphs on n vertices is the following: We first run the randomized algorithm Approx that is guaranteed in Theorem 4.1 with parameter $\epsilon/2$, to produce a graph G^* that is a (d, s) union and is $\epsilon/2$-close to G with high probability. Having a full knowledge of G^*, without further queries to G, Algorithm T_Π checks if G^* is $\epsilon/2$-close to Π_n. It accepts if the answer is yes, and reject otherwise.

Note that T_π is oblivious of Π when performing the queries to G. Once the queries are made to G and G^* is obtained, a test for any property can be run (in parallel, say).

To analyse the error probability, assume that G^* is indeed $\epsilon/2$-close to G, as asserted by Theorem 4.1. This happens with probability at least 0.9. Now if G has Π, then T_Π would accept, because $G \in \Pi_n$, and G^* is $\epsilon/2$-close G, which makes it $\epsilon/2$-close to Π_n. On the other hand, if T_Π accepts on account of finding an $H \in \Pi_n$, and such that G^* is $\epsilon/2$-close to H, then by triangle inequality G is ϵ-close to Π_n. Thus the error probability is bounded by 0.1. ◀

Proof of Theorem 4.2. Let G, H be two outerplanar graphs on which we want to ϵ-test isomorphism. The ϵ-test will be as follows: It will first run the randomized algorithm Approx, as guaranteed by Theorem 4.1, to produce a G^* that is (d, s)-union, with distance parameter $\epsilon/2$.

It will then consider the graph property of n-vertex graphs $\Pi(G^*)$ to be the following property: *the input graph is $\epsilon/2$-close to G^**. By Theorem 4.3, there is an $\epsilon/2$-tester T' for

$\Pi(G^*)$. We just run T' on H, accept if it does and reject if it rejects. Note that the query complexity is just doubled. Note also that G^* and hence T' are not known in advance, but this does not matter, as we do not need to worry about the time complexity.

To analyse the success probability, assume that G^* is $\epsilon/2$-close to G, which is asserted to happen with probability 0.9. Now, assume that H is isomorphic to G, than G^* is also $\epsilon/2$-close to H, and hence H has property $\Pi(G^*)$. Thus, test T' will indeed accept H with probability at least 0.9. On the other hand, assume that T' accepts H, then with probability 0.9, H is $\epsilon/2$ close to $\Pi(G^*)$. Then, by the triangle inequality it is ϵ-close to G as required. The total error is hence bounded by the events, that either G^* is not $\epsilon/2$-close to G or that T' errs. As both are bounded by 0.1, the total error probability is at most 0.2. ◀

Proof of Theorem 4.4 (Sketch). The proof in this case is somewhat more involved than the previous theorems. The basic idea, as in [9] is that if G, H have the same local view, then applying on both the sampler of Theorem 4.1, one will get identical (or close enough), approximations G^*, H^* respectively, as the approximation is done based on the information in the local views, which is identical for both graphs. However, there was a difficulty in this argument, even in [9] for bounded-degree hyperfinite graphs. To understand what the difficulty is, let us start to formalize the proof.

Let D be as in Theorem 4.1, and $R = f(D)$ be a constant depending on D, to be defined later. Let $q = poly(\log n)$ be the number of queries asserted by the algorithm Sampler that is used in the proof of Theorem 4.1, for some fixed ϵ.

Let $V = V(G)$, and $V' = V(H)$. Assume that the R-views of G is identical to the R-views of H. Hence, we can fix a $1-1$ map $\phi : V \mapsto V'$ so that every v is mapped to $v' = \phi(v)$ such that the R-disc of v is identical to the R-disc of v'. Our aim is to simulate the process of the sampler on H, by observing its run in G: that is, if the sampler on G is making some q queries to discs around vertices $v_1, \ldots, v_q$, we will aim to use queries on identical discs of their images $v'_1, \ldots, v'_q$ in H, with the hope that the output graph G^* produced by the sampler on G will be identical to the output graph H^* obtained from H.

The first problem is that the sampler on G might be successful in G, in respect of obtaining a good approximation G^*, using the queries $v_1, \ldots, v_q$, while the output graph H^* obtained from H on the corresponding sequence $v'_1, \ldots, v'_q$ might not be a good approximation of H. However, as both process are assured to be successful with high probability, for most sequences $v_1, \ldots v_q$, the processes on G and the corresponding one on H are both going to be successful: on G with queries to discs of $v_1, \ldots v_q$, and on H with queries to discs of $v'_1 = \phi(v_1), \ldots, v'_q = \phi(v_q)$. It is not clear however, that they will produce the same approximation although they seemingly see the same view, due to the following reason.

The sampler needs to makes some q i.i.d queries, to components in G^i, in order to approximate $lfreq(G^i)$. Concentrate first on G^0 (which is an identical case to that considered in [9]). The sampler makes queries to a sequence $v_1, \ldots v_q$ (on which as explained above, we may assume it will succeed), and explore the D-disc around each v_i on which it can run the local partition oracle on the graph restricted to low-degree vertices, in order to define the component of each v_i in G^0. Now suppose that v_i is now being queried and that a vertex u in the D-disc(v_i) is also present in the D-disc(v_j), for some previously queried vertex v_j. Since the partition must be consistent, the neighbourhood around u that is discovered when v_j was queried, is the same when viewed exploring the disc around v_i. However, from the view point in H, while v'_j, v'_i have isomorphic discs of the appropriate size as v_j, v_i respectively, they do not have to share a vertex $u' = \phi(u)$. Namely, the $u \in D\text{-}disc(v_j)$ is mapped to u' that is not necessarily in $D\text{-}disc(v_i)$.

The argument in [9] addressed this issue is the following way: since the degree is bounded by some constant d, a situation as above (that for two random D-discs there is a non-empty intersection) has very low probability, and hence will not occur on most random sequences.

Here this is not correct any more, as the degree of the roots may be as high as $\Omega(n)$ in higher layers. Then it could happen that any two discs around such high-degree vertices do intersect. To get rid of this problem assume first that there are no edges with two endpoints that are high-degree, both in G and H. This may be assumed, as for the map ϕ above, vertices are mapped to vertices with isomorphic discs even after deleting edges between two high degree vertices. Further assume that the set of edges $M = \{(u, y)| \ u \in V^l, \ y \in V^h\}$ is of size $|M| \geq \frac{\epsilon n}{q^2 \log n}$ (otherwise, there is no problem as no high-degree vertex is likely to be seen at all as a root (in the first item in phase 3 of the sampler - In addition, in this case the graph is close to a d-bounded degree, and hence the argument above for d bounded degree graphs implies that with high probability the sampler will produce an identical approximation for both G and H, when simulated as explained above.

Recall that our sampler makes a total of q queries (which is $poly(\log n)$ in our case). At the top level, it makes some queries to random edges in M (in phase 3 first item, sub-item (a)), in order to make independent queries to at most q randomly chosen root vertices in G^i for every level i, and explore the component in G^i under such roots, by random sampling.

Consider the bad case when a low-degree vertex v might be found while randomly exploring components formed by two such high-degree roots y and y'. Assume that y' is chosen after y, and that the random neighbours of y that are queried are $u_1, \ldots u_r$, where $r \leq q$. Then while forming the (d, s)-components in $G[V^l]$ for $u_1, \ldots, u_r$, they together involves querying a total of at most q low-degree vertices (v being among them). Further, these vertices have a total of $r \leq q$ edges that are queried and whose end points are high-degree vertices other than y. Call these edges "bad" with respect to y.

Now, for v to be queried while exploring y', the same should happen with y', i.e., v should be among the total of at most q queried edges once the above exploration is done with y'. However, v will not be queried while exploring y', if no bad edge with respect to y is queried while exploring y', and if the D-discs in $G[V^l]$ around the at most q queries from y to low-degree vertices do not intersect the $D\text{-}disc(v)$ in $G[V^l]$. Hence, when y' is chosen by a random edge, if none of its at most q queried random neighbours of y' are bad edges w.r.t. y, then v will not be queried while exploring y', due to the first reason, and conditioned on that, the D-discs around these random vertices will also be disjoint from $D\text{-}disc(v)$ as $D\text{-}disc(v)$ contains only a very small (constant) number of vertices .

Therefore, for y' such that $deg(y') \geq q^5$, while exploring y', encountering a low-degree vertex u that was encountered earlier while exploring from another such y will happen with probability at most $1/q^3$.

For y' such that $deg(y') \leq q^5$, the layer containing y' has a high mass (more than $n/\log^2 n$) only if that layer contains at least $\frac{n}{sq^5 \log^2 n}$ such high-degree vertices, where $s = O(d^2/\epsilon^2)$ is the bound set by the partitioning algorithm on the component size. Now, for such y', if none of the low-degree neighbours u of y' has the edge (u, y') that is bad w.r.t y, again exploration from y' will not query a v that was queried while exploring y. As there are at most q bad edges w.r.t. y, at most q high-degree vertices y's will have one of these bad edges incident on them, and hence the probability of picking such a y' for exploration is at most $\frac{s \cdot q^6 \log^2 n}{n}$ which is extremely low.

Over all, combining the two cases, the probability that for (y, y') as above, a common v will be queried is lower than $1/q^3$, and hence by the union bound on all possible q^2 pairs, with very high probability, for no pair (y, y') a common vertex is queried.

Assuming that indeed for no pair (y, y') of high-degree vertices, there is a common low-degree vertex that is queried, the local views that are sampled for each root in $G^i, i \geq \alpha$ are distinct.

Hence we can couple the two sampling processes, the one for G and that for H in a consistent way, so to have the same views, and therefore will produce G^* and H^* which are identical.

Thus, if we run the sampler of Theorem 4.8 with parameter $\epsilon/2$ on G and H, it is ensured that with high probability the runs are successful and it produces G^* and H^* respectively with $dist(G, G^*) \leq \epsilon n/2$, $dist(H, H^*) \leq \epsilon n/2$ and moreover $G^* = H^*$. Therefore, we have $dist(G, H) = \epsilon$, as desired.

Finally, let us consider what is the disc radius needed to ensure that the above will indeed occur. Note that for low-degree vertices, we only need D-discs around them to be able simulate the sampler behaviour from the local information. For high-degree vertices, some $r \leq q$ random queries to neighbours are being made in the disc around them. While $q = poly(\log n)$ and not constant, note that all the possibly q such queries, are done to neighbours (namely, vertices of distance 1), from such high-degree vertices. Further queries are done in $G[V^l]$ to simulate the local partition on low-degree vertices, with an occasional query that discovers a high-degree vertex, but, in which case, no further exploration is done from this high-degree vertex. Hence taking $2D$-discs is enough to simulate the sampler behaviour.

We avoid further details in this version. ◀

5 Discussion

Our results are another step towards understanding the theory of property testing in the sparse graph model, and mainly for restricted subfamilies of planar graphs. Yet the main questions in this area are still open:

- Which graph properties are testable with sub-linear query complexity?
- Is it true that if two n vertices planar graphs H, G have their D-local views identical (for some large enough constant D), then the graphs are ϵ-close to be isomorphic? Is this true for bounded tree-width graphs ?

We note that the above questions are open, even for the class of 2-outerplanar graphs.

References

1 Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. *SIAM J. Comput.*, 39(1):143–167, 2009. doi:10.1137/060667177.

2 Itai Benjamini, Oded Schramm, and Asaf Shapira. Every minor-closed property of sparse graphs is testable. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, 2008*, pages 393–402, 2008. doi:10.1145/1374376.1374433.

3 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.

4 Avinatan Hassidim, Jonathan A. Kelner, Huy N. Nguyen, and Krzysztof Onak. Local graph partitions for approximation and testing. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, pages 22–31, 2009. doi:10.1109/FOCS.2009.77.

5 Hiro Ito. Every property is testable on a natural class of scale-free multigraphs. *CoRR*, abs/1504.00766, 2015. URL: http://arxiv.org/abs/1504.00766.

6 Mitsuru Kusumoto and Yuichi Yoshida. Testing forest-isomorphism in the adjacency list model. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Proceedings, Part I*, pages 763–774, 2014. doi:10.1007/978-3-662-43948-7_63.

7 Reut Levi and Dana Ron. A quasi-polynomial time partition oracle for graphs with an excluded minor. *ACM Trans. Algorithms*, 11(3):24:1–24:13, 2015. doi:10.1145/2629508.

8 Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980.

9 Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM J. Comput.*, 42(3):1095–1112, 2013. doi:10.1137/120890946.

10 Krzysztof Onak. New sublinear methods in the struggle against classical problems. PhD Thesis, Massachusetts Institute of Technology, 2010.

11 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

12 Christian Sohler. Private Communication, 2015.

A Missing Proofs of Section 3

Proof of Lemma 3.2. We will prove the lemma by induction on $|V| = n$. Assume that $s, t \in V$ and $|V| \geq 4$. Since G is 2-connected and $(s,t) \in E$, then all the vertices are on a simple path between s and t. Enumerate the vertices along this path $v_1 = s, v_2, \dots v_n = t$. Let $i < n$ be the largest such that $(s, v_i) \in E$, and let $j > 1$ be the smallest such that $(v_j, t) \in E$. Since G is outerplanar it follows that $i \leq j$, therefore either $i \leq \lceil \frac{n}{2} \rceil$ or $\lceil \frac{n}{2} \rceil \leq j \leq |V|$. Assume w.l.o.g that $i \leq \lceil n/2 \rceil$ and let $V_1 = \{s, v_2, ..., v_i\}$. Note that $G[V_1]$ is outerplanar, with (s, v_i) an edge on the outer face. If $i = 2$, then $C_1 = \{(s, v_i)\}$ will separate s from v_i in $G[V_1]$. If $i > 2$, $G[V_1]$ is 2-connected and by induction hypothesis, there is an edge-cut C_1 separating between s and v_i in $G[V_1]$, with $|C_1| \leq \lfloor \log(\lceil n/2 \rceil + 1) \rfloor$. It is easy to see that $C_1 \cup (s,t)$ is a $\{s,t\}$-multiway cut in G, of size as claimed. ◀

▶ **Lemma 1.1.** *Let G be 2-connected outerplanar graph. For any pair of vertices $s, t \in V(G)$, $c(s,t) \leq 2(\log(|V(G)| + 1))$.*

Proof of Lemma 1.1. Let G be 2-connected outerplanar graph, and $s, t \in V(G)$. Since G is outerplanar, then all vertices of G are on the unique Hamiltonian cycle C of G. We may assume that s, t are not neighbours on C, as otherwise, Lemma 3.2 immediately implies the result. Hence, C defines two vertex disjoint paths, from s to t: $P_1 = (s, v_1, \dots, v_k = t)$, and $P_2 = (s, u_1, \dots, u_\ell = t)$. Let $i \leq k$ be the largest such that $(s, v_i) \in E$ and $j < \ell$ the largest such that $(s, u_j) \in E$. Then $G_1 = G[\{s, v_1, \dots, v_i\}]$ is outerplanar with (s, v_i) on its outer face. If $i = 1$, then $C_1 = \{(s, v_i)\}$ will separate s from v_i in G_1. Otherwise, G_1 is 2-connected and by Lemma 3.2, there exist an edge cut C_1 in G_1 separating s and v_i with $|C_1| \leq \log(|V(G)| + 1)$. Similarly, $G_2 = G[\{s, u_1, \dots, u_j\}]$ is outerplanar with (s, u_j) on its outer face and has an edge cut C_2 separating s and u_j with $|C_2| \leq \log(|V(G)| + 1)$. It is easy to see that $C_1 \cup (s,t)$ is an edge-cut in G separating s and t, of size as claimed. ◀

▶ **Lemma 1.2.** *Let $G(V, E)$ be a connected outerplanar graph and $U, W \subsetneq V$ be disjoint subsets of vertices. Suppose that $|U| \geq 2$ and U is an independent set in G. Then there exists an edge cut of size $2\log(2|W| + 1)$ in $G[W \cup U]$ separating some two points in U.*

Proof of Lemma 1.2. Let $G(V,E)$ be a connected outerplanar graph and $U, W \subsetneq V$ be disjoint subsets of vertices. Suppose $G[W]$ is connected and U is an independent set in G such that every vertex in U is a neighbour of some vertex in W and $|U| > 1$.

First consider the case when $G[W \cup U]$ is 2-connected. Since U is an independent set, then in the Hamiltonian cycle that is the boundary of the outer face of $G[W \cup U]$, between every two vertices of U, there must be at least one vertex from W. Hence $|U| \le |W|$, which implies that $|U \cup W| \le 2|W|$. Take any two arbitrary vertices $u_1, u_2 \in U$. By Lemma 1.1, there exists an edge-cut C of size at most $2(\log(|U|+|W|+1)) \le 2(\log(2|W|+1))$ separating u_1 and u_2 in $G[W \cup U]$.

The same argument as above applies also when $G[W \cup U]$ is not 2-connected, and there is a block $\mathcal{B}$ of $G[W \cup U]$ that contains two vertices from U. Hence we may assume that every block of G contains at most one vertex form U.

Let $\mathcal{B}$ be a block of $G[W \cup U]$ containing a single vertex u from U. Let u' be another vertex in U, which by the reasoning above, is in another block $\mathcal{B}'$ of $G[W \cup U]$. Let x be the cut-vertex in $\mathcal{B}$ which is a separation point between u and u' in $G[W \cup U]$. In this case, $|V(\mathcal{B})| \le |W| + 1$ and by Lemma 1.1, u can be separated from x in $\mathcal{B}$ by removing at most $2(\log(|W|+2))$ edges. Such a cut also separates u from u'. ◀

Proof of Corollary 3.3. We will prove this by induction on $|U|$. Let $t = 2\log(2|W|+1)$. The proof is trivial when $|U| = 1$. If $|U| > 1$, by Lemma 1.2 an edge cut C of size t exists in $G[W \cup U]$ that separates a subset $S \subsetneq U$ from $U \setminus S$ where $1 \le |S| < |U|$. Let us now consider $G_1 = G[W \cup S]$ and $G_2 = G[W \cup (U \setminus S)]$. By the induction hypothesis there is a S-multiway cut C_1 of size $(|S|-1)t$ in the graph G_1 and there is a $(U \setminus S)$ multiway cut of size C_2 of size $(|(U \setminus S)|-1)t$ in G_2. Taking $C \cup C_1 \cup C_2$ we obtain a U-multiway cut in G of size $(|U|-1)t$. ◀

Proof of Claim 3.4. For every vertex $b \in B$ remove all but exactly two edges. Hence we get a subgraph G_1 of G in which $deg(b) = 2$ for every $b \in B$. Hence, every $b \in B$ form a simple path of length 2 in G_1. Replace each such path by a single edge; this is equivalent to the contraction of a single edge in the neighbourhood of every $b \in B$. As a result we get a multigraph G_2 that is a minor of G, and hence outerplanar. In this multigraph, the number of parallel edges between two vertices is at most two, as otherwise in G, there would have been a $K_{2,3}$ minor, which is a contradiction. Hence $|E(G_2)| \le 4|V(G_2)|$.

However, by construction, every edge $e \in E(G_2)$ corresponds to a vertex $b \in B$, with a $1-1$ correspondence. Further $V(G_2) = A$. Hence the claim follows. ◀

B Algorithm Approx

Let $G = (V,E)$ be an outerplanar graph, $\epsilon > 0$ the error parameter, and let G' be the (d,s)-union graph obtained by *Algorithm 1*, for $\epsilon' = \epsilon/10$ (and d accordingly).

The purpose of the algorithm Approx is to estimate the γ-layered graph G^* that is $\epsilon/10$-close to G' (and hence $\epsilon/5$-close to G). That is, for each $i \in [L]$ to give an approximation to n_i the number of high-degree components in G^i, and $lfeq[i]$ the frequency vector of G^i, where L is as defined in Definition 4.5.

Algorithm Approx runs a main algorithm Sampler that samples high-degree vertices of G' accoring to a distribution in which all root vertices in the same layer are sampled with the same probability. Sampler runs another algorithm, Sampler2 that estimates the degree and the component frequency of the sampled vertex in order for Sampler to be able to update n_i and $lfreq[i]$ accordingly.

We now present the algorithm SAMPLER2, which given an outerplanar graph $G(V, E)$, and a vertex y, approximates $freq_{G'}(y)$ and $deg_{G'}(y)$.

Let $q = poly(\log n)$ (e.g., the reader may take $q = 10\log^3 n$ to get the right magnitude, the exact value will not be defined here).

Algorithm SAMPLER2(y)

y is a vertex. The output is an estimate $\widetilde{deg_{G'}}(y)$, and an estimate $\widetilde{freq_{G'}}(y)$ for $deg_{G'}(y)$ and $freq_{G'}(y)$ respectively.

1. Obtain $deg_G(y)$ by one query. If $deg(y) \leq d$ stop; y is not a root of a (d, s)-rooted component in G'.
   Otherwise, let $c = 0$ (c will count the number of discovered (d, s)-components that are connected to y). Let $Freq(y)$ be the all-zero vector of dimension $f(d, s)$.
2. Repeat independently for q times: Choose a random low-degree neighbour $u \in_R V^l \cap N(y)$.

   Look at the component of u in $G[V^l]$ (by applying Levi-Ron, as needed), take the multiway cut, and finally see if y is still in the same component as u in G' which is the graph obtained by the simulation of Algorithm 1 locally on v (some edges adjacent to y might be deleted, due to the multiway cut procedure, and due to the deletion of edges between two high-degree vertices).
   If y is still in the same component as u in G^*, increment c. Also, depending on the type of the (d, s) component containing u and connected to y, increment the corresponding coefficient of $Freq(y)$.
3. Take $\widetilde{deg_{G'}}(y) = deg_G(y) \cdot c/q$, and $\widetilde{freq_{G'}}(y) = Freq(y)/c$.

It is easy to see that SAMPLER2 will estimate well the required parameters for y such that $deg_{G'}(y)$ is high enough. This motivates the following definition.

▶ **Definition 2.1.** Let $Bad = \{y | \ deg_{G'}(y) \leq deg_G(y)/(20 \log n)\}$.

We present the following lemma without proof.

▶ **Lemma 2.2.** *Let G' be the graph obtained from a graph G by the partition, and $y \notin Bad$. Then for the output of* SAMPLER2 *it holds that $Pr[|\widetilde{deg_{G'}}(y) - deg_{G'}(y)| \geq \epsilon^2 \cdot deg_{G'}(y)/100] \leq 1/\log^5 n$, and $Pr[|freq(y) - \widetilde{freq}(y)| \geq \epsilon^2/100] \leq 1/\log^5 n$.*

The proof is a standard application of Chernoff-Hoefding bound and will not be presented here.

We now present the algorithm SAMPLER, which given an outerplanar graph $G(V, E)$ as input, returns a high-degree vertex in G'. Among high-degree vertices in any fixed layer i, the probability of obtaining each one will be roughly the same and the total probability of returning any one of the roots in layer i will be roughly proportional to $n_i \gamma^i$, where n_i is the number of high-degree vertices in layer i.

Algorithm SAMPLER

1. Sample uniformly at random, a vertex $v \in V(G)$, if $deg_G(v) \geq d$ reject.
2. Otherwise, if $deg_G(v) \leq d$, query all neighbours of v and if for all $y \in N(v)$, $deg_G(y) \leq d$ reject.
3. Otherwise, let $N^h(v) = \{y | \ deg_G(y) > d\}$ and let $deg^h(v) = |N^h(v)|$. Choose uniformly a random member $y \in N^h(v)$. Discard y and reject with probability $1 - \frac{deg^h(v)}{d}$. Otherwise (with probability $\frac{deg^h(v)}{d}$), we will consider y to be a candidate for a (random) root of G'.

4. Run Sampler2(y). If y not rejected, check if v is a neighbour of y in G' by simulating locally Algorithm 1, (as is done in Sampler2, for other random neighbours of y). If not reject.
5. With probability $\gamma^i/\widetilde{deg_{G'}}(y)$, return y, where i is the largest for which $\gamma^i \leq \widetilde{deg_{G'}}$.

We claim that for each layer of G^*, the distribution that Sampler2(y) indices on its roots is (nearly) uniform. Moreover, if there is enough "mass" in $G^i, i = \alpha, \dots L$ then Sampler will produce a random root of G' (in some layer) w.h.p.

Before stating the corresponding lemma, we note that what appears to be a trivial sampling is not correct. Namely, choosing a random vertex $y \in V(G)$ and returning y if its degree is high enough is not likely to succeed, as it might be the case that the number of roots in G' is very small (possibly 1), while their degree is very high. In such a case, a random sampling will not find a random root, while the influence of the small number of roots on the structure of G' (in terms of distance to G), is very high.

▶ **Lemma 2.3.** *Suppose that $\sum_{y \in roots(G')} \deg_{G'}(y) \geq \epsilon n/\log n$ then* Sampler *produces a random $y \in roots(G')$ distributed uniformly on each layer of G^*, with probability at least $1/\log^2 n$.*

Proof. We do not present the full proof of the lemma in this version. We will only prove that Sampler induces the uniform probability on each layer of G^* and that with high probability it will output such y.

Indeed, consider a fixed $y \in roots(G')$. The only way y is going to be accepted is if a v that is selected in step 1 is one of the $deg_{G'}(y)$ neighbours of y in G'. Denote this set of neighbours of y by N'. Each such v is chosen with probability $1/n$ at step 1, and will pass step 2. Now, conditioned on specific v chosen, the probability that y is chosen in step 3 and not discarded is $\frac{1}{deg^h(v)} \cdot \frac{deg^h(v)}{d} = \frac{1}{d}$. Finally, if y is chosen at step 3, and conditioned on the event that Sampler2(y) accepts y and estimates $\deg_{G'}(y)$ correctly, which happen with probability $1 - 1/\log^5 n$, y will be accepted with probability $\gamma^i/\widetilde{deg_{G'}}(y)$. Altogether, the probability of returning y is:

$$Prob(\text{Sampler returns } y) = \sum_{v \in N'} \frac{1}{n} \cdot \frac{1}{d} \cdot \frac{1}{\widetilde{deg_{G'}}(y)} = \frac{1}{nd} \cdot \frac{\gamma^i \cdot deg_{G'}(y)}{\widetilde{deg_{G'}}(y)}.$$

Assuming the estimate is as good as we needed, this probability is very close to $\frac{\gamma^i}{nd}$ and hence to uniform on each layer of G^*, as it is essentially independent of y but just on the layer.

Finally, let i be such that $n_i \cdot \gamma^i = \Omega(n/\log n)$, namely, such that the ith layer in G^* has a large mass. The probability some y in the i layer of G^* is accepted the sum above for all y in the layer which is just $n_i \cdot \frac{\gamma^i}{nd} \geq \frac{\epsilon}{d \log n}$. By our choice of d, the proof is concluded. ◀

Algorithm Approx is now self evident: it runs Sampler for $poly(\log n)$ times to generate enough random roots to hit all significant layers G^i. Would there be no vertices in Bad, the estimate would clearly be correct, by Chernoff-Hoefding bounds. The effect of vertices in Bad can be shown to be small, as the total number connected to vertices in Bad is small. We avoid further details in this version.

The Condensation Phase Transition in the Regular k-SAT Model[*]

Victor Bapst[1] and Amin Coja-Oghlan[2]

1 **Mathematics Institute, Goethe University, Frankfurt, Germany**
bapst@math.uni-frankfurt.de
2 **Mathematics Institute, Goethe University, Frankfurt, Germany**
acoghlan@math.uni-frankfurt.de

Abstract

Much of the recent work on phase transitions in discrete structures has been inspired by ingenious but non-rigorous approaches from physics. The physics predictions typically come in the form of distributional fixed point problems that mimic Belief Propagation, a message passing algorithm. In this paper we show how the Belief Propagation calculation can be turned into a rigorous proof of such a prediction, namely the existence and location of a condensation phase transition in the regular k-SAT model.

1998 ACM Subject Classification G.2.1 Combinatorics, G.3 Probability and Statistics, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases random k-SAT, phase transitions, Belief Propagation, condensation

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.22

1 Introduction

1.1 Background and motivation

Over the past three decades the study of random constraint satisfaction problems has been driven by ideas from statistical mechanics [3, 24, 25]. The physics ideas have since had a substantial impact on algorithms, coding theory and combinatorics [12, 15, 18, 19, 20, 21, 30]. The striking feature of the physics work is that it is based on one generic but non-rigorous technique called the *cavity method* that can be applied almost mechanically [23]. Its centerpiece is the *Belief Propagation* message-passing algorithm. By contrast, rigorous studies have largely been case-by-case.

This state of affairs begs the question of whether the Belief Propagation calculations can be put on a rigorous basis directly. This is precisely the thrust of the present paper. We show how the physics calculations can be turned into a proof in a highly non-trivial and somewhat representative case. We expect that this approach generalises to many other alike problems. Specifically, we determine the precise *condensation phase transition* in the random regular k-SAT model. The existence of such a phase transition in a wide variety of models is one of the key predictions of the cavity method [22] and its impact on algorithmic as well as information-theoretic question can hardly be overstated [29, 32]. For example, the condensation phenomenon has a bearing on the performance of message-passing

[*] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 278857–PTCC.

algorithm such as Belief Propagation guided decimation [29] as well as on statistical inference problems [5]. Moreover, the regular k-SAT problem shares many of the key properties of the better-known model where clauses are simple chosen uniformly and independently; in particular, a condensation phase transition is expected to occur in that model as well [22]. The proof builds upon on our abstract results [6] on the "regularity method" for discrete probability measures and the connection to spatial mixing properties.[1]

1.2 The regular k-SAT model

Consider variables $x_1, \ldots, x_n$ that may take the values 'true' or 'false', represented by $+1$ and -1. If $\Phi = \Phi_1 \wedge \cdots \wedge \Phi_m$ is a k-CNF formula, then we define a function $E_\Phi : \{\pm 1\}^n \to \{0, 1, \ldots, m\}$ on the set of truth assignments by letting $E_\Phi(\sigma)$ be the number of violated clauses. In physics jargon, E_Φ is called the *Hamiltonian.* Further, we define the *Gibbs measure* at "inverse temperature" $\beta \geq 0$ by letting

$$\sigma \in \{\pm 1\}^n \mapsto \exp(-\beta E_\Phi(\sigma))/Z_\Phi(\beta) \qquad \text{where} \qquad Z_\Phi(\beta) = \sum_{\sigma \in \{\pm 1\}^n} \exp(-\beta E_\Phi(\sigma)) \quad (1)$$

is called the *partition function.* Thus, the Gibbs measure is a probability measure on the cube $\{\pm 1\}^n$.

As β gets larger the mass of the Gibbs measure shifts to assignments that violated fewer clauses. Ultimately, if we let $\beta \to \infty$, then the Gibbs measure concentrates on the maximally satisfying assignments. Hence, by tuning β we can "scan" the landscape that the function E_Φ defines on the cube $\{\pm 1\}^n$. Among other things, grasping this landscape is key in order to study the performance of local search algorithms such as Simulated Annealing or the Metropolis process, which attempt to descend from a random starting point to a global minimum. For instance, if E_Φ is riddled with local minima, local search algorithms are bound to get trapped, while they might be efficient on a nice "convex" landscape [1, 9, 25].

It turns out that the key quantity upon which the study of the Hamiltonian hinges is the partition function. Therefore, we aim to calculate $Z_{\mathbf{\Phi}}(\beta)$ on a random k-CNF formula $\mathbf{\Phi}$. There are several natural probability distributions on k-SAT formulas. The one that we study here is perhaps the simplest non-trivial example, namely the *regular k-SAT model* [28]. It comes with two integer parameters $k \geq 3$ and $d > 1$, which is even. For n such that $2k$ divides dn we let $\mathbf{\Phi} = \mathbf{\Phi}_{d,k}(n)$ signify a uniformly random k-SAT formula with $m = dn/(2k)$ clauses of length k over $x_1, \ldots, x_n$ such that each variable x_i occurs precisely $d/2$ times as a positive literal x_i and precisely $d/2$ times as a negative literal $\neg x_i$.[2] For k exceeding a certain constant k_0 there is an explicitly known critical degree $d_{k-\mathrm{SAT}}$, the *satisfiability threshold*, where satisfying assignments cease to exist in a typical $\mathbf{\Phi}$ [12][3]. While the exact formula is cumbersome, asymptotically we have

$$d_{k-\mathrm{SAT}}/k = 2^k \ln 2 - k \ln 2/2 - (1 + \ln 2)/2 + o_k(1), \qquad (2)$$

where $o_k(1)$ hides a term that tends to 0 in the limit of large k. Since $Z_{\mathbf{\Phi}}(\beta)$ scales exponentially with n, we consider

$$\phi_{d,k} : \beta \in (0, \infty) \mapsto \lim_{n \to \infty} \frac{1}{n} \mathbb{E}[\ln Z_{\mathbf{\Phi}}(\beta)] \qquad (3)$$

[1] The present paper builds upon the arXiv version of [6] because the version that appeared in the proceedings of RANDOM 2015 contained a critical error.

[2] The regular k-SAT model shares many of the properties of the better known model where m clauses are chosen uniformly and independently but avoids the intricacies that result from degree fluctuations.

[3] We have $\liminf \mathbb{P}[\mathbf{\Phi} \text{ is satisfiable}] > 0$ if $d < d_{k-\mathrm{SAT}}$ and $\lim \mathbb{P}[\mathbf{\Phi} \text{ is satisfiable}] = 0$ if $d > d_{k-\mathrm{SAT}}$.

with the log *inside* the expectation and the expectation is over $\mathbf{\Phi}$. The existence of the limit follows from the interpolation method [10]. Moreover, Azuma's inequality implies that $\ln Z_{\mathbf{\Phi}}(\beta)$ concentrates about $\mathbb{E}[\ln Z_{\mathbf{\Phi}}(\beta)]$ for any d, k, β.

We call $\beta_0 \in (0, \infty)$ *smooth* if there exists $\varepsilon > 0$ such that the function

$$\beta \in (\beta_0 - \varepsilon, \beta_0 + \varepsilon) \mapsto \phi_{d,k}(\beta)$$

admits an expansion as an absolutely convergent power series around β_0. Otherwise a *phase transition* occurs at β_0.[4] Thus, with d fixed we aim to investigate the effect of tuning β.

Results. According to the "cavity method" for certain values of d close to the satisfiability threshold $d_{k-\mathrm{SAT}}$ there occurs a so-called *condensation phase transition* at a certain critical $\beta_{\mathrm{cond}}(d, k) > 0$ [22]. The main result of this paper proves this conjecture for k exceeding a certain constant k_0. Let us postpone the precise definition of $\beta_{\mathrm{cond}}(d, k)$ for just a moment.

▶ **Theorem 1.** *There exists $k_0 \geq 3$ such that for all $k \geq k_0$, $d \leq d_{k-\mathrm{SAT}}$ there is $\beta_{\mathrm{cond}}(d, k) \in (0, \infty]$ such that all $\beta \in (0, \beta_{\mathrm{cond}}(d, k))$ are smooth. If $\beta_{\mathrm{cond}}(d, k) < \infty$, then there occurs a phase transition at $\beta_{\mathrm{cond}}(d, k)$.*

We will see momentarily that $\beta_{\mathrm{cond}}(d, k) < \infty$ for d exceeding a specific critical degree $d_{\mathrm{cond}}(k) < d_{k-\mathrm{SAT}}$. Theorem 1 is the first rigorous result to identify the precise critical "inverse temperature" in a random constraint satisfaction problem, apart perhaps from the far simpler case of the stochastic block model [26].

Let us take a look at $\beta_{\mathrm{cond}}(d, k)$. As most predictions based on the cavity method, $\beta_{\mathrm{cond}}(d, k)$ results from a *distributional fixed point problem*, i.e., a fixed point problem on the space of probability measures on the open unit interval $(0, 1)$. This fixed point problem derives mechanically from the physicists' "1RSB cavity equations" [23]. Specifically, writing $\mathcal{P}(0, 1)$ for the set of probability measures on the unit interval, we define two maps

$$\mathcal{F}_{k,d,\beta} : \mathcal{P}(0,1) \to \mathcal{P}(0,1), \qquad \hat{\mathcal{F}}_{k,d,\beta} : \mathcal{P}(0,1) \to \mathcal{P}(0,1)$$

as follows. Given $\pi \in \mathcal{P}(0, 1)$ let $\eta = (\eta_1, \ldots, \eta_{k-1}) \in (0, 1)^{k-1}$ be a random $k-1$-tuple drawn from the distribution $(\hat{z}(\eta)/\hat{Z}(\pi))\,\mathrm{d}\bigotimes_{j=1}^{k-1} \pi(\eta_j)$, where

$$\hat{z}(\eta) = 2 - (1 - \exp(-\beta)) \prod_{j<k} \eta_j \qquad \text{and} \qquad \hat{Z}(\pi) = \int \hat{z}(\eta)\mathrm{d}\bigotimes_{j<k} \pi(\eta_j). \tag{4}$$

Then $\widehat{\mathcal{F}}_{k,d,\beta}(\pi)$ is the distribution of $(1 - (1 - \exp(-\beta)) \prod_{i=1}^{k-1} \eta_i)/\widehat{z}(\eta)$. Similarly, given $\hat{\pi} \in \mathcal{P}(0, 1)$ draw $\hat{\eta} = (\hat{\eta}_1, \ldots, \hat{\eta}_{d-1})$ from $(z(\hat{\eta})/Z(\hat{\pi}))\mathrm{d}\bigotimes_{j=1}^{k-1} \hat{\pi}(\hat{\eta}_j)$, where

$$z(\widehat{\eta}) = \prod_{j<d/2} \widehat{\eta}_j \prod_{j\geq d/2} (1 - \widehat{\eta}_j) + \prod_{j<d/2} (1 - \widehat{\eta}_j) \prod_{j\geq d/2} \widehat{\eta}_j, \qquad Z(\widehat{\pi}) = \int z(\widehat{\eta})\mathrm{d}\bigotimes_{j<k} \widehat{\pi}(\widehat{\eta}_j). \tag{5}$$

Then $\mathcal{F}_{k,d,\beta}(\widehat{\pi})$ is the distribution of $(\prod_{j<d/2} \widehat{\eta}_j \prod_{j\geq d/2}(1 - \widehat{\eta}_j))/z(\widehat{\eta})$. Further, call a distribution $\pi \in \mathcal{P}(0, 1)$ *skewed* if

$$\pi[(\exp(-k^{0.9}\beta), 1 - \exp(-k^{0.9}\beta))] < 2^{-0.9k}.$$

[4] This is the usual physics definition of a "phase transition". The motivation is that the non-analyticity of $\phi_{d,k}$ indicates a qualitative change. For illustration, observe that the fraction of vertices in the largest component of the Erdős-Rényi random graph is non-analytic at average degree one.

▶ **Proposition 2.** *Let $d_-(k) = d_{k-\mathrm{SAT}} - k^5$ and $\beta_-(k,d) = k\ln 2 - 10\ln k$. The map $\mathcal{G}_{k,d,\beta} = \mathcal{F}_{k,d,\beta} \circ \widehat{\mathcal{F}}_{k,d,\beta}$ has a unique skewed fixed point $\pi^\star_{k,d,\beta}$, provided that $k \geq k_0$, $d \in [d_-(k), d_{k-\mathrm{SAT}}]$ and $\beta > \beta_-(k,d)$.*

To extract $\beta_{\mathrm{cond}}(d,k)$, let $\nu_1, \ldots, \nu_k, \hat{\nu}_1, \ldots, \hat{\nu}_d$ be independent random variables such that the ν_i have distribution $\pi^\star_{k,d,\beta}$ and the $\hat{\nu}_i$ have distribution $\widehat{\mathcal{F}}_{k,d,\beta}(\pi^\star_{k,d,\beta})$. Setting

$$z_1 = \prod_{j \leq d/2} \hat{\nu}_j \prod_{j > d/2} (1 - \hat{\nu}_j) + \prod_{j \leq d/2} (1 - \hat{\nu}_j) \prod_{j > d/2} \hat{\nu}_j, \qquad z_2 = 1 - (1 - \exp(-\beta)) \prod_{j \leq k} \nu_j$$

and $z_3 = \nu_1 \hat{\nu}_1 + (1 - \nu_1)(1 - \hat{\nu}_1)$, we let

$$\mathcal{F}(k,d,\beta) = \ln \mathbb{E}\left[z_1\right] + \frac{d}{k} \ln \mathbb{E}\left[z_2\right] - d \ln \mathbb{E}[z_3], \tag{6}$$

$$\mathcal{B}(k,d,\beta) = \frac{\mathbb{E}\left[z_1 \ln z_1\right]}{\mathbb{E}\left[z_1\right]} + \frac{d}{k} \frac{\mathbb{E}\left[z_2 \ln z_2\right]}{\mathbb{E}\left[z_2\right]} - d \frac{\mathbb{E}[z_3 \ln z_3]}{\mathbb{E}[z_3]}. \tag{7}$$

Finally, with the usual convention that $\inf \emptyset = \infty$ we let

$$\beta_{\mathrm{cond}}(k,d) = \begin{cases} \infty & \text{if } d < d_-(k), \\ \inf\{\beta > \beta_-(k,d) : \mathcal{F}(k,d,\beta) < \mathcal{B}(k,d,\beta)\} & \text{if } d \in [d_-(k), d_{k-\mathrm{SAT}}]. \end{cases}$$

We proceed to highlight a few consequences of Theorem 1 and its proof. The following result shows that $\beta_{\mathrm{cond}}(d,k) < \infty$, i.e., that a condensation phase transition occurs, for degrees d strictly below the satisfiability threshold.

▶ **Corollary 3.** *If $k \geq k_0$, then $d_{\mathrm{cond}}(k) = \min\{d > 0 : \beta_{\mathrm{cond}}(d,k) < \infty\} < d_{k-\mathrm{SAT}} - \Omega(k)$.*

Furthermore, the following corollary shows that the so-called "replica symmetric solution" predicted by the cavity method yields the correct value of $\phi_{d,k}(\beta)$ for $\beta < \beta_{\mathrm{cond}}(d,k)$.

▶ **Corollary 4.** *If $k \geq k_0$, $d \leq d_{k-\mathrm{SAT}}$ and $\beta < \beta_{\mathrm{cond}}(d,k)$, then $\phi_{d,k}(\beta) = \mathcal{F}(k,d,\beta)$.*

Corollary 4 opens the door to studying the "shape" of the Hamiltonian $E_{\mathbf{\Phi}}$ for $\beta < \beta_{\mathrm{cond}}(d,k)$, a necessary step towards studying, e.g., the performance of local search algorithms. Specifically, Corollary 4 enables us to bring the "planting trick" from [1] to bear so that we can analyse typical properties of samples from the Gibbs measure.

Finally, complementing Corollary 4, the following result shows that $\mathcal{F}(k,d,\beta)$ overshoots $\phi_{d,k}(\beta)$ for $\beta > \beta_{\mathrm{cond}}(d,k)$.

▶ **Corollary 5.** *If $k \geq k_0$, $d \leq d_{k-\mathrm{SAT}}$ and $\beta > \beta_{\mathrm{cond}}(d,k)$, then there is $\beta_{\mathrm{cond}}(d,k) < \beta' < \beta$ such that $\phi_{d,k}(\beta') < \mathcal{F}(k,d,\beta')$.*

2 Techniques and related work

Admittedly, the definition of $\beta_{\mathrm{cond}}(k,d)$ is not exactly simple. For instance, even though the fixed point distribution from Proposition 2 stems from a discrete problem, it is a continuous distribution on $(0,1)$. Yet the analytic formula (6) is conceptually *far* simpler than the combinatorial definition of $\phi_{d,k}$. Indeed, we are going to see in Section 3 that the fixed point problem can be understood in terms of a branching process, i.e., a random infinite tree.

The proof of Theorem 1 builds upon an abstract result from [6] that, roughly speaking, breaks the study of the partition function down into two tasks. First, to prove that the Gibbs measure induced by a random formula $\hat{\mathbf{\Phi}}$ chosen from a reweighted probability distribution, the "planted model", enjoys the *non-reconstruction property*, a spatial mixing property. Second, to analyse Belief Propagation on $\hat{\mathbf{\Phi}}$. The technical contribution of the present work is to tackle these two problems in a fairly generic way. In fact, we expect that the proof strategy extends to other problems. A concrete example that springs to mind is the Potts antiferromagnet on a random graph, which is intimately related to the information-theoretic threshold in the "stochastic block model" with multiple classes [5]. While conceptually the proof strategy allows us to turn the Belief Propagation calculation into a rigorous theorem in a fairly direct way, the technical challenge of actually analysing the relevant Belief Propagation fixed point in a completely rigorous manner remains.

The overall proof strategy bears some resemblance to the work of Mossel, Neeman and Sly [26] on the "stochastic block model", but the details are quite different. Roughly speaking, the stochastic block model can be viewed as a planted version of the minimum bisection problem and the problem is to recover the labels that were used to generate the graph. The proof from [26] that this is not possible up to a certain point relies on non-reconstruction as well. Moreover, the contiguity estabished in [26] can be viewed as a condensation result, albeit with the much simpler interactions of the stochastic block model. In particular, the "condensation threshold" is merely given by a quadratic equation rather than a distributional fixed point equation.

The predictions of the "cavity method" typically come as distributional fixed points but there are only few proofs that establish such predictions rigorously. The one most closely related to the present work is [7] on condensation in random graph coloring. It determines the critical average degree d for which condensation starts to occur with respect to the number of proper k-colorings of the Erdös-Rényi random graph. This corresponds to taking $\beta \to \infty$ in (3). This simplifies the problem substantially because in the limit "frozen variables" emerge that are fixed deterministically to one specific value. Other previous results on condensation gave only approximate answers [8, 13, 14].

Interestingly, determining the satisfiability threshold on $\mathbf{\Phi}$ is conceptually easier than identifying the condensation threshold [12]. This is because the local structure of the regular random formula is essentially deterministic, namely a tree comprising of clauses and variables in which every variable appears $d/2$ times positively and $d/2$ times negatively. In effect, the satisfiability threshold is given by a fixed point problem on the unit interval, rather than on the space of probability measures on the unit interval. Similar simplifications occur in other regular models [17, 16]. By contrast, we will see in Section 3 that the condensation phase transition hinges on the reweighted distribution $\hat{\mathbf{\Phi}}$ with a genuinely random local structure.

Recent work on the k-SAT threshold in uniformly random formulas [12, 11] and in particular the breakthrough paper by Ding, Sly and Sun [18], also harnessed the Belief/Survey Propagation calculations and [18] verified the prediction in terms of the corresponding distributional fixed point problem.[5] In the uniformly random model a substantial technical complication is posed by variables of exceptionally high degree. While [12, 11, 18] apply the second moment method to a random variable whose construction is guided by Belief/Survey Propagation, here we employ Belief Propagation in the direct way enabled by [6].

Talagrand [31] and, by means of a different argument, Panchenko [27] studied the k-SAT model on uniformly random formulas in the "high-temperature" (i.e., small β) case. Specifically, with d the average degree of a variable, [27, 31] require that $\min\{4\beta, 1\}(k-1)d < 1$. This range of parameters is well below the conjectured condensation phase transition [22].

[5] Survey Propagation can be viewed as a Belief Propagation applied to a modified constraint satisfaction problem [23].

3 Proof outline

We assume that $k \geq k_0$ for a large enough constant k_0 and that $d < d_{k-\mathrm{SAT}}$.

3.1 Two moments do not suffice

The default approach to studying $\phi_{d,k}(\beta)$ would be the venerable "second moment method" [3]. Cast on a logarithmic scale, if

$$\limsup_{n\to\infty} \frac{1}{n} \ln \mathbb{E}[Z_{\mathbf{\Phi}}(\beta)^2] \leq \lim_{n\to\infty} \frac{2}{n} \ln \mathbb{E}[Z_{\mathbf{\Phi}}(\beta)], \qquad \text{then} \tag{8}$$

$$\phi_{d,k}(\beta) = \lim_{n\to\infty} \frac{1}{n} \ln \mathbb{E}[Z_{\mathbf{\Phi}}(\beta)]. \tag{9}$$

Thus, if (8) holds, then we can "swap the log and the expecation". Unsurprisingly, calculating $\ln \mathbb{E}[Z_{\mathbf{\Phi}}(\beta)]$ is fairly easy (see (11) below).

From a bird's eye view, both the physics intuition and the second moment are all about the geometry of the Gibbs measure of $\mathbf{\Phi}$ at a given $\beta \in (0,\infty)$. Indeed, according to the physics picture the condensation point $\beta_{\mathrm{cond}}(k)$ should be the supremum of all $\beta > 0$ such that w.h.p. for two random assignments $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2 \in \{\pm 1\}^n$ chosen from the Gibbs measure we have $|\boldsymbol{\sigma}_1 \cdot \boldsymbol{\sigma}_2| = o_n(n)$, i.e., $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2$ are about orthogonal [22]. This is a necessary condition for the success of the second moment method as well [2, 4], which may instil hopes that (8) might hold for β right up to $\beta_{\mathrm{cond}}(d,k)$. In fact, (8) holds if either d or β is relatively small.

▶ **Lemma 6.** *For $d \leq d_{k-\mathrm{SAT}}$ and $\beta > 0$ let $q \in (0,1)$ be the unique solution to the equation*

$$1 - (1 - \exp(-\beta))q^k = 2(1-q). \tag{10}$$

Then

$$\frac{1}{n} \ln \mathbb{E}\left[Z_\beta(\mathbf{\Phi})\right] \sim \ln 2 + \frac{d}{k} \ln\left(1 - (1-\exp(-\beta))q^k\right) + \frac{d}{2} \ln(4q(1-q)). \tag{11}$$

Furthermore, if either $d \leq d_-(k)$ or $\beta \leq \beta_-(k,d)$ then (8) is true.

However, for d close to $d_{k-\mathrm{SAT}}$ and β near $\beta_{\mathrm{cond}}(d,k)$ the second moment method fails. Formally, if d is such that $\beta_{\mathrm{cond}}(d,k) < \infty$, then there exists $\beta' < \beta_{\mathrm{cond}}(d,k)$ such that (8) is violated for all $\beta \in (\beta', \beta_{\mathrm{cond}}(d,k))$.

3.2 Quenching the average

To understand what goes awry we turn the second moment into a first moment with respect to reweighted distribution. Specifically, the *planted model* is the random pair $(\hat{\mathbf{\Phi}}, \hat{\boldsymbol{\sigma}})$ chosen from the distribution

$$\mathbb{P}\left[(\hat{\mathbf{\Phi}}, \hat{\boldsymbol{\sigma}}) = (\hat{\Phi}, \hat{\sigma})\right] = \frac{\exp(-\beta E_{\hat{\Phi}}(\hat{\sigma}))}{(dn)! \cdot \mathbb{E}[Z_{\mathbf{\Phi}}(\beta)]}. \tag{12}$$

Thus, the probability that $(\hat{\Phi}, \hat{\sigma})$ comes up is proportional to $\exp(-\beta E_{\hat{\Phi}}(\hat{\sigma}))$. Further, the probability that a specific formula $\hat{\Phi}$ comes up equals $\mathbb{P}[\hat{\mathbf{\Phi}} = \hat{\Phi}] = Z_\beta(\hat{\Phi})/((dn)! \cdot \mathbb{E}[Z_\beta(\mathbf{\Phi})])$, proportional to the partition function. In effect,

$$\mathbb{E}[Z_{\mathbf{\Phi}}(\beta)^2] = \mathbb{E}[Z_{\mathbf{\Phi}}(\beta)] \cdot \mathbb{E}[Z_{\hat{\mathbf{\Phi}}}(\beta)]. \tag{13}$$

Hence, in light of (11) computing the second moment is equivalent to calculating $\mathbb{E}[Z_{\hat{\mathbf{\Phi}}}(\beta)]$.

In fact, the second moment calculation from the proof of Lemma 6 reveals that $\mathbb{E}[Z_{\hat{\mathbf{\Phi}}}(\beta)]$ is dominated by two distinct contributions. First, assignments that are more or less orthogonal to $\hat{\boldsymbol{\sigma}}$ yield a term of order $\mathbb{E}[Z_{\mathbf{\Phi}}(\beta)]$. The second contribution is from $\boldsymbol{\sigma}$ close to $\hat{\boldsymbol{\sigma}}$; say, $\boldsymbol{\sigma}\cdot\hat{\boldsymbol{\sigma}} \geq n(1-2^{-k/10})$. Geometrically, this reflects the fact that the "planted assignment" $\hat{\boldsymbol{\sigma}}$ sits in a "valley" of the Hamiltonian $E_{\hat{\mathbf{\Phi}}}$ w.h.p. The valleys are officially called *clusters* and we let

$$\mathcal{Z}_{\hat{\mathbf{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta) = \sum_{\sigma\in\{\pm1\}^n} \mathbf{1}\{\boldsymbol{\sigma}\cdot\hat{\boldsymbol{\sigma}} > n(1-2^{-k/10})\}\exp(-\beta E_{\hat{\mathbf{\Phi}}}(\sigma)). \tag{14}$$

be the Gibbs-weighted *cluster size*. Hence, (13) shows that the second moment method functions iff $\mathbb{E}[\mathcal{Z}_{\hat{\mathbf{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)] \leq \mathbb{E}[Z_{\mathbf{\Phi}}(\beta)]$.

But for d close to $d_{k-\mathrm{SAT}}$ and $\beta > \beta_{\mathrm{cond}}(d,k)$ we have $\mathbb{E}[\mathcal{Z}_{\hat{\mathbf{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)] \geq \exp(\Omega(n))\mathbb{E}[Z_{\mathbf{\Phi}}(\beta)]$. In other words, the expected cluster size blows up. At a second glance, this is unsurprising. For the cluster size scales exponentially with n and is therefore prone to large deviations effects. To suppress these we ought to work with $\mathbb{E}[\ln \mathcal{Z}_{\hat{\mathbf{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)]$ instead of $\mathbb{E}[\mathcal{Z}_{\hat{\mathbf{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)]$. A similar issue (that the expected cluster size explodes) occurred in earlier work on condensation [7, 8, 13, 14]. Indeed, borrowing the idea of a truncated second moment method from these papers, we can reduce the computation of $\phi_{d,k}(\beta)$ to the problem of determining $\mathbb{E}[\ln \mathcal{Z}_{\hat{\mathbf{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)]$.

▶ **Lemma 7.** *Equation* (9) *holds iff*

$$\limsup_{n\to\infty} n^{-1}\mathbb{E}[\ln \mathcal{Z}_{\hat{\mathbf{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)] \leq \lim_{n\to\infty} n^{-1}\ln\mathbb{E}[Z_{\mathbf{\Phi}}(\beta)]. \tag{15}$$

Hence, we are left to calculate $\mathbb{E}[\ln \mathcal{Z}_{\hat{\mathbf{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)]$, the "quenched average" in physics jargon. As we saw the log and the expectation do not commute. In such cases, computing the quenched average is notoriously difficult, certainly well beyond the reach of elementary methods. Tackling this problem is the main achievement of this paper; recall the expressions from (6)–(7).

▶ **Proposition 8.** *Assume that $d \in [d_-(k), d_{k-SAT}]$ and $\beta > \beta_-(k,d)$. Then*

$$\lim_{n\to\infty}\frac{1}{n}\mathbb{E}[\ln \mathcal{Z}_{\hat{\mathbf{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)] = \mathcal{B}(k,d,\beta), \qquad \textit{while} \qquad \lim_{n\to\infty}\frac{1}{n}\ln\mathbb{E}[Z_{\mathbf{\Phi}}(\beta)] = \mathcal{F}(k,d,\beta).$$

We observe that Theorem 1 is immediate from Lemma 6, Lemma 7 and Proposition 8.

3.3 Non-reconstruction

To calculate the quenched average we are going to have to understand the typical internal structure of the cluster in the planted model. According to the physicists "1-step replica symmetry breaking picture", the restriction of the Gibbs measure to the cluster should enjoy a spatial mixing property called *non-reconstruction*. In particular, the truth values assigned to variables that are "far apart" are predicted to be asymptotically independent.

If non-reconstruction holds, then a general result from [6] reduces the computation of the quenched average to determining the marginals of the restricted Gibbs distribution, which we are going to calculate via Belief Propagation.

Formally, by the restriction of the Gibbs measure to the cluster we mean the probability distribution on $\{\pm1\}^n$ defined by

$$\sigma \in \{\pm1\}^n \mapsto \mathbf{1}\{\boldsymbol{\sigma}\cdot\hat{\boldsymbol{\sigma}} > n(1-2^{-k/10})\}\exp(-\beta E_{\hat{\mathbf{\Phi}}}(\sigma))/\mathcal{Z}_{\hat{\mathbf{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta). \tag{16}$$

For a random variable $X(\boldsymbol{\sigma})$ we denote the average with respect to (16) by

$$\langle X(\boldsymbol{\sigma})\rangle' = \langle X(\boldsymbol{\sigma})\rangle'_{\hat{\boldsymbol{\Phi}},\hat{\boldsymbol{\sigma}},\beta} = \frac{1}{\mathcal{Z}_{\hat{\boldsymbol{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)} \sum_{\sigma\in\{\pm1\}^n} \mathbf{1}\{\boldsymbol{\sigma}\cdot\hat{\boldsymbol{\sigma}} > n(1-2^{-k/10})\}\exp(-\beta E_{\hat{\boldsymbol{\Phi}}}(\sigma)).$$

Further, to define a metric we set up a bipartite graph whose vertices are the clauses and variable of $\hat{\boldsymbol{\Phi}}$. Each clause is adjacent to all the variables that it contains. Then the distance between two variables or clauses is, of course, the length of a shortest path in the graph.

We can now state the non-reconstruction condition. For a variable x, an integer $\ell \geq 0$ and $\tau \in \{\pm1\}^n$ let $\nabla(\hat{\Phi}, x, \ell, \tau)$ be the set of all $\chi \in \{\pm1\}^n$ such that $\chi(y) = \tau(y)$ for all y at distance at least 2ℓ from x in $\hat{\boldsymbol{\Phi}}$. Then

$$\left\langle \boldsymbol{\sigma}(x)|\nabla(\hat{\boldsymbol{\Phi}}, x, \ell, \tau)\right\rangle'$$

is the average of the truth value of x once we condition on the event that the truth values of all variables at distance at least 2ℓ from x are given by the "boundary condition" τ. Thus, we inspect the distribution of the truth value of x given the faraway variables.

The non-reconstruction condition requires that for most variables x, $\langle \boldsymbol{\sigma}(x)|\nabla(\hat{\boldsymbol{\Phi}}, x, \ell, \boldsymbol{\tau})\rangle'$ is close to $\langle \boldsymbol{\sigma}(x)\rangle'$ in expectation with respect to a boundary condition $\boldsymbol{\tau}$ that is itself chosen randomly from (16). Formally, $(\hat{\boldsymbol{\Phi}}, \hat{\boldsymbol{\sigma}})$ has the *non-reconstruction property w.h.p.* if for any $\varepsilon > 0$ there is $\ell > 0$ such that

$$\lim_{n\to\infty} \mathbb{P}\left[\frac{1}{n}\sum_{i=1}^{n}\left\langle\left|\langle\boldsymbol{\sigma}(x_i)\rangle' - \left\langle\boldsymbol{\sigma}(x_i)|\nabla(\hat{\boldsymbol{\Phi}}, x_i, \ell, \boldsymbol{\tau})\right\rangle'\right|\right\rangle' < \varepsilon\right] = 1. \tag{17}$$

▶ **Proposition 9.** *Assume that $d \in [d_-(k), d_{k-SAT}]$ and $\beta > \beta_-(k,d)$. Then $(\hat{\boldsymbol{\Phi}}, \hat{\boldsymbol{\sigma}})$ has the non-reconstruction property w.h.p.*

Together with [6, Theorems 4.4–4.5] Proposition 9 reduces the computation of the quenched average to the problem of computing the marginals under the measure (16). Specifically, $\lim_{n\to\infty}\frac{1}{n}\mathbb{E}[\ln \mathcal{Z}_{\hat{\boldsymbol{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)]$ is given by an expression called the *Bethe free energy* that is a function of the vector $(\langle\boldsymbol{\sigma}(x_i)\rangle')_{i=1,\ldots,n}$ of marginals only.[6] The Bethe free energy originally comes from the physicists cavity method [23, ch. 14].

3.4 A branching process

Hence, we are left to calculate the marginals of (16). Due to the correlation decay guaranteed by the non-reconstruction property, the marginals are governed by the local structure of the formula $\hat{\boldsymbol{\Phi}}$. To facilitate the marginal computation, we are going to condition on the event that the planted assignment $\hat{\boldsymbol{\sigma}} = \mathbf{1}$ is the all-ones vector; this is without loss of generality because under the planted model (12) $\hat{\boldsymbol{\sigma}}$ is uniformly distributed.

Of course, in $\hat{\boldsymbol{\Phi}}$ each variable occurs $d/2$ times positively and $d/2$ times negatively. But the distribution of the signs with which the variables occur in the clauses is non-trivial. We are going to describe it via a branching process with four types: variable nodes of type ±1 and clause nodes of type ±1. Starting from a single variable node r, the process is defined as follows; let $q \in (0,1)$ be the solution to (10).

[6] Stirctly speaking, Proposition 9 and [6, Theorems 4.4 and 4.5] merely imply that the Bethe free energy is an upper bound on $\lim \frac{1}{n}\mathbb{E}[\ln \mathcal{Z}_{\hat{\boldsymbol{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)]$. To obtain the matching lower bound it is necessary to consider another version of the planted model, see the appendix for details.

BR1: For the root r let $b_{r,\uparrow} = 1$ with probability $1-q$ and $b_{r,\uparrow} = -1$ with probability q.

BR2: Suppose that x is a variable node of type $b_{x,\uparrow} = \pm 1$. Then x has $d-1$ children, which are clause nodes. Specifically, $\frac{d}{2}-1$ children a are clause nodes of type $b_{a,\uparrow} = b_{x,\uparrow}$, and the remaining $d/2$ children are clause ndoes of type $b_{a,\uparrow} = -b_{x,\uparrow}$.

BR3: Suppose that a is a clause node of type $b_{a,\uparrow} = 1$. Then a has $k-1$ children in total, which are variable nodes. Specifically, $X_a = \mathrm{Bin}(k-1, 1-q)$ children have type 1, and the remaining $k-1-X_a$ children have type -1.

BR4: Finally, suppose that a is a clause node of type $b_{a,\uparrow} = -1$. Then a has $k-1$ children, which are variable nodes and
- with probability $\exp(-\beta)q^{k-1}/(1-(1-\exp(-\beta))q^{k-1})$ all children have type -1,
- otherwise $Y_a = \mathrm{Bin}_{\geq 1}(k-1, 1-q)$ children have type 1 and the others have type -1.

Let us write $\boldsymbol{T}_\infty$ be the random infinite tree generated by this branching process (including the type assignment $b_{\,\cdot\,,\uparrow}$). Moreover, let $\mathcal{T}_\infty$ be the set of all possible outcomes.

We can think of $\boldsymbol{T}_\infty$ as an infinite k-SAT formula in which all variables other than r appear $d/2$ times positively and $d/2$ times negatively. Namely, for each clause node a we define a Boolean clause whose variables are the parent variable node of a and the $k-1$ children of a. The sign with which the parent x of a occurs in a is precisely $b_{a,\uparrow}$, the type of a. Thus, a contains the literal x if $b_{a,\uparrow} = 1$ and the literal $\neg x$ otherwise. Similarly, each child y of a occurs with sign $b_{y,\uparrow}$.

The root of $\boldsymbol{T}_\infty$ has degree $d-1$ rather than d. This will be useful to set up the Belief Propagation equations below, but to describe the local structure of $\hat{\boldsymbol{\Phi}}$ we actually need a tree in which the root has degree d. Thus, let $\boldsymbol{T}'_\infty$ be the infinite tree defined just as above except that the root has $d/2$ children of type $+1$ and $d/2$ children of type -1. Further, let $\mathcal{T}'_\infty$ be the set of all possible outcomes of this process.

The tree $\boldsymbol{T}'_\infty$ captures the local structure of $\hat{\boldsymbol{\Phi}}$. More precisely, for a formula Φ and a variable x let $\Delta^l_\Phi x$ be the sub-formula obtained from Φ by deleting all clauses and variables at distance at least l from x. Additionally, for a specific formula φ let

$$\rho_{\hat{\boldsymbol{\Phi}}}(\varphi) = \frac{1}{n}\left|\left\{x : \Delta^{2\ell+1}_{\hat{\boldsymbol{\Phi}}} x \cong \varphi\right\}\right|$$

be the fraction of variables x of $\hat{\boldsymbol{\Phi}}$ whose depth-2ℓ neighborhood is isomorphic to φ.

▶ **Lemma 10.** *For all ℓ, φ we have* $\mathbb{E}\left|\rho_{\hat{\boldsymbol{\Phi}}}(\varphi) - \mathbb{P}\left[\Delta^{2\ell+1}\boldsymbol{T}'_\infty \cong \varphi\right]\right| = O(n^{-1/2}\ln n)$.

In light of Lemma 10 we can study the marginals of (16) by way of the random tree $\boldsymbol{T}'_\infty$. Specifically, we are going construct a map $\mathcal{T}'_\infty \to \mathcal{P}(\{\pm 1\})$ that yields a probability measure on $\{\pm 1\}$ for each tree such that the marginal of a variable x is close to the conditional expectation of this map given the depth-2ℓ neighborhood $\Delta^{2\ell+1}_{\hat{\boldsymbol{\Phi}}} x$ for large enough ℓ. It will emerge that this map is intimately related to the fixed point problem from Proposition 2. To construct the map $\mathcal{T}'_\infty \to \mathcal{P}(\{\pm 1\})$ we employ Belief Propagation; for a detailed introduction to Belief Propagation and the physics intuition behind it see [23].

3.5 Belief Propagation

Fix some integer $\ell \geq 1$. Viewing the tree $T \in \mathcal{T}_\infty$ as a k-SAT formula as above, we let $V_{2\ell}$ be the set of all variable nodes at distance at most 2ℓ from the root of T and let $F_{2\ell}$ be the set of all clause nodes at distance at most 2ℓ from the root. Further, let $\partial V_{2\ell}$ be the set of all variable nodes of T at distance exactly 2ℓ from the root. Belief Propagation starts from a *boundary condition* $\partial\nu : \partial V_{2\ell} \to \mathcal{P}(\{\pm 1\})$, $x \mapsto \partial\nu_x$ that assigns each x a probability distribution on $\{\pm 1\}$. The *Belief Propagation messages* induced by the boundary condition $\partial\nu$ on T are the (unique) families

$$(\nu_{x,\uparrow}^{T,\partial\nu})_{x\in V_{2\ell}}, \qquad (\widehat{\nu}_{a,\uparrow}^{T,\partial\nu})_{a\in F_{2\ell}}$$

of probability measures on $\{\pm 1\}$ determined by the following three conditions. For a node u of T let $\partial_\downarrow u$ be the set of children.

BP1: For all $x \in \partial V_{2\ell}$ we have $\nu_{x,\uparrow}^{T,\partial\nu} = \partial\nu_x$.

BP2: For all $x \in V_{2\ell} \setminus \partial V_{2\ell}$ and $s \in \{-1, 1\}$,

$$\nu_{x,\uparrow}^{T,\partial\nu}(s) = \frac{\prod_{a\in\partial_\downarrow x} \widehat{\nu}_{a,\uparrow}^{T,\partial\nu}(s)}{\sum_{s'\in\{-1,1\}} \prod_{a\in\partial_\downarrow x} \widehat{\nu}_{a,\uparrow}^{T,\partial\nu}(s')}. \tag{18}$$

BP3: For all $a \in F_{2\ell}$ and $s \in \{-1, 1\}$,

$$\widehat{\nu}_{a,\uparrow}^{T,\partial\nu}(s) = \frac{\sum_{s_a\in\{-1,1\}^{\partial a}} \mathbf{1}\{s_x = s\}\, \psi_a(s_a) \prod_{y\in\partial_\downarrow a} \nu_{y,\uparrow}^{T,\partial\nu}(s_y)}{\sum_{s_a\in\{-1,1\}^{\partial a}} \psi_a(s_a) \prod_{y\in\partial_\downarrow a} \nu_{y,\uparrow}^{T,\partial\nu}(s_y)}. \tag{19}$$

Algorithmically, all messages can be calculated bottom-up from the boundary $V_{2\ell}$. The "result" of the Belief Propagation calculation on T given a certain boundary condition is the message emanating from the root:

$$\nu_T^{\partial\nu} = \nu_{r,\uparrow}^{T,\partial\nu}.$$

The fixed point distribution from Proposition 2 can be obtained organically by running Belief Propagation on $\boldsymbol{T}_\infty$. Indeed, define $\partial\nu^{(0)} : \partial V_{2\ell} \to \mathcal{P}(\{\pm 1\})$ by $\partial\nu_x^{(0)}(1) = 1$ for all $x \in \partial V_{2\ell}$ and let $\nu_T^{(2\ell)} = \nu_{r,\uparrow}^{T,\partial\nu^{(0)}}$.

▶ **Proposition 11.** *Assume that $d_-(k) < d \le d_{k-SAT}$ and $\beta > \beta_-(k, d)$. The sequence $(\nu_{\boldsymbol{T}_\infty}^{(2\ell)})_{\ell\ge 1}$ converges almost surely to a limit $\nu_{\boldsymbol{T}_\infty}^\star$. Moreover, $\pi_{k,d,\beta}^\star$ is the distribution of the random variable $\nu_{\boldsymbol{T}_\infty}^\star(-b_{r,\uparrow})$.*

We define the Belief Propagation messages $\nu_{T'}^{(2\ell)}$ for the trees $T' \in \mathcal{T}'_\infty$ in which the root r has degree d exactly as we did above. Of course, the calculation of the messages $\nu_{T'}^{(2\ell)}$ is closely related to that of the messages $\nu_T^{(2\ell)}$ for $T \in \mathcal{T}_\infty$; after all, the only difference occurs at the root. The proof of Proposition 11 shows that the Belief Propagation recurrence enjoys certain contraction properties. In combination with the non-reconstruction property we thus obtain an asymptotic formula for the marginals of the distribution (16).

▶ **Proposition 12.** *Assume that $d_-(k) < d \le d_{k-SAT}$ and $\beta > \beta_-(k, d)$. The sequence $(\nu_{\boldsymbol{T}'_\infty}^{(2\ell)})_{\ell\ge 1}$ converges almost surely to a limit $\nu_{\boldsymbol{T}'_\infty}^\star$. Moreover,*

$$\lim_{\ell\to\infty} \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\left|\langle \mathbf{1}\{\boldsymbol{\sigma}(x_i) = 1\}\rangle'_{\hat{\boldsymbol{\Phi}},\mathbf{1},\beta} - \mathbb{E}[\nu_{\boldsymbol{T}'_\infty}^\star(1) | \Delta^{2\ell+1}\boldsymbol{T}'_\infty \cong \Delta_{\hat{\boldsymbol{\Phi}}}^{2\ell+1} x]\right| = 0$$

Plugging the asymptotic marginals from Proposition 11 into the Bethe free energy formula, we obtain an expression for the quenched average $\lim \frac{1}{n}\mathbb{E}[\ln \mathcal{Z}_{\hat{\boldsymbol{\Phi}},\hat{\boldsymbol{\sigma}}}(\beta)]$. Due to the inherent connection between $\nu_{\boldsymbol{T}_\infty}^\star$ and $\nu_{\boldsymbol{T}'_\infty}^\star$, a (tedious) bit of calculus reveals that this formula can be expressed in terms of the fixed point distribution $\pi_{k,d,\beta}^\star$. The Bethe free energy formula then morphs into the expression $\mathcal{B}(k, d, \beta)$ from (7). In the course of this we also find that (6) matches the "annealed average" $\lim \frac{1}{n} \ln \mathbb{E}[Z_{\boldsymbol{\Phi}}(\beta)]$. Thus Proposition 8 follows.

In the following two sections we outline the two key parts of the proof in some more detail, namely the proof of the non-reconstruction property and the analysis of Belief Propagation on the random tree.

4 Non-reconstruction

We assume that $d \in [d_-(k), d_{k-SAT}]$ and that $\beta \geq \beta_-(k,d)$. Let $c_\beta = 1 - \exp(-\beta)$.

To prove Proposition 9 we exhibit six deterministic conditions that entail the non-reconstruction property. First, a formula Φ on variables $V = \{x_1, \ldots, x_n\}$ satisfies property **ℓ-Local Structure** if

ℓ-Local Structure: for all trees T of height $2\ell + 2$ we have

$$|\rho_\Phi(T) - \mathbb{P}\left[\Delta^{2\ell+3}\boldsymbol{T}'_\infty \cong T\right]| \leq n^{-0.49}.$$

In words, the empirical distribution of the depth-$2\ell + 2$ neighbourhoods is close to the distribution of the random tree $\boldsymbol{T}'$.

The second condition reads

Cycles: The formula Φ contains $o(\sqrt{n})$ cycles of length at most $\sqrt{\ln n}$.

To state the third condition we identify a large "well-behaved" bit of the random formula that we call the *core*. Similar constructions have been used extensively in prior work on random constraint satisfaction problems (e.g., [1, 7, 8]). Let $\partial_{\pm 1} x$ be the set of clauses where the variable x appears as a positive/negative literal and, conversely, let $\partial_{\pm 1} a$ be the set of variables that appear in clause a positively/negatively. Now, the λ-*core* of Φ (in symbols: $\mathrm{Core}_\lambda(\Phi)$) is the largest set W of variables such that all $x \in W$ satisfy the following conditions.

CR1: there are at least $\lambda^{-1}k^{0.99}$ clauses $a \in \partial_1 x$ such that $\partial_1 a = \{x\}$.
CR2: there are no more than 10λ clauses $a \in \partial x$ such that $|\partial_{-1} a| = k$.
CR3: for any $1 \leq l \leq k$ the number of $a \in \partial_{-1} x$ such that $|\partial_1 a| = l$ is bounded by $\lambda k^{l+3}/l!$.
CR4: there are no more than $\lambda k^{3/4}$ clauses $a \in \partial_1 x$ such that $|\partial_1 a| = 1$ but $\partial a \not\subset W$.
CR5: there are no more than $\lambda k^{3/4}$ clauses $a \in \partial_{-1} x$ such that $|\partial_{-1} a| < k$ and $|\partial_1 a \setminus W| \geq |\partial_1 a|/4$.

The λ-core is well-defined; for if W, W' satisfy the above conditions, then so does $W \cup W'$. Further, if $\lambda < \lambda'$, then $\mathrm{Core}_\lambda(\Phi) \subset \mathrm{Core}_{\lambda'}(\Phi)$. The formula Φ has the property **λ-Core** if

λ-Core: $|\mathrm{Core}_\lambda(\Phi)| \geq (1 - 2^{-0.95k})n$.

We are going to identify a large set $V_{\mathrm{good}} \subset V$ of variables that are very likely to be set to one under a typical assignment $\boldsymbol{\sigma}$ chosen from $\langle \cdot \rangle'_{\Phi,\beta}$. As a first attempt we might try $V_{\mathrm{good}} = \mathrm{Core}_{1/2}(\Phi)$. However, the conditions **CR1**–**CR5** are not quite strong enough to enable an estimate of the $\langle \cdot \rangle'_{\Phi,\beta}$-marginals. For instance, if the marginals of most of the neighbors of a given vertex $x \in \mathrm{Core}_{1/2}(\Phi)$ go astray, x will likely follow suit. Yet the variables x in the core such that $\langle \boldsymbol{\sigma}(x) \rangle'_{\Phi,\beta}$ is "small" must clump together. Formally, we say that a set $S \subset V$ is λ-*sticky* if for all $x \in S$ one of the following conditions holds.

ST1: There are at least $\lambda k^{3/4}$ clauses $a \in \partial_1 x$ such that $\partial_1 a = \{x\}$ and $\partial_{-1} a \cap S \neq \emptyset$.
ST2: Yhere are at least $\lambda k^{3/4}$ clauses $a \in \partial_{-1} x$ such that $|\partial_{-1} a| < k$ and $|\partial_1 a \cap S| \geq |\partial_1 a|/4$.

Further, Φ satisfies the property **λ-Sticky** if

λ-Sticky: Φ has no λ-sticky set of size between $2^{-0.95k}n$ and $2^{-k/20}n$.

The condition **Sticky** ensures that for $\lambda \in \{1/2, 1\}$ there is a unique maximal λ-sticky set $S_\lambda(\Phi) \subset \mathrm{Core}_\lambda(\Phi)$ of size $S_\lambda(\Phi) \leq 2^{-0.1k}n$. Indeed, if $S, S' \subset \mathrm{Core}_\lambda(\Phi)$ are two λ-sticky sets of size at most $2^{-0.1k}n$, then $S \cup S'$ is sticky as well. Consequently, **Sticky** guarantees that $|S \cup S'| \leq 2^{-0.95k}n$. In fact, this argument shows that $S_\lambda(\Phi) \leq 2^{-0.95k}n$.

Further, the next condition reads

Gap: $\left\langle \mathbf{1}\{\boldsymbol{\sigma}\cdot\mathbf{1} < (1-2^{-k/3})n\}\right\rangle' \leq \exp(-\Omega_n(n))$.
Hence, comparing the above with (14), we realise that **Gap** requires that assignments $\boldsymbol{\sigma}$ with $1-2^{-k/10} < \boldsymbol{\sigma}\cdot\mathbf{1}/n < 1-2^{-k/3}$ contribute little to the cluster size.

Finally, we come to the seventh and last condition. A variable $x \in V$ is $(\varepsilon, 2\ell)$-*cold* if the following two conditions are satisfied. Write $\partial^{2\ell}x$ for the set of all variables at distance exactly 2ℓ from x.
CD1: The sub-formula $T = \Delta^{2\ell+1}x$ is a tree.
CD2: If $\boldsymbol{\tau} : \partial^{2\ell}x \to \{\pm 1\}$ is a random assignment such that independently for all $y \in \partial^{2\ell}x$,

$$\boldsymbol{\tau}(y) = \begin{cases} -1 & \text{if } y \notin \mathrm{Core}_1(\Phi) \cup S_1(\Phi) \\ (-1)^{\mathrm{Be}(\exp(-k^{0.9}\beta))} & \text{otherwise} \end{cases},$$

then

$$\mathbb{E}\left[\max\left\{\left|\nu_T^{(2\ell)}(1) - \langle \mathbf{1}\{\boldsymbol{\sigma}(x)=1\}|\nabla(\Phi,x,\ell,\tau)\rangle'\right| : \tau \geq \boldsymbol{\tau}\right\}\right] \leq \varepsilon. \tag{20}$$

In words, suppose that we choose a random "boundary condition" $\boldsymbol{\tau}$ such that all y at distance 2ℓ from x that do not belong to the core are set to -1 and all y in the core are set to -1 with probability $\exp(-k^{0.9}\beta)$ independently. Then an adversary comes along and obtains τ from $\boldsymbol{\tau}$ maliciously by setting $\tau(y) = 1$ for a few y such that $\boldsymbol{\tau}(y) = -1$. (The adversary is not allowed to make changes in the opposite direction.) Then (20) requires that the spin $\boldsymbol{\sigma}(x)$ given the boundary condition τ be close to the Belief Propagation marginal $\nu_T^{(2\ell)}(1)$. Of course, the expectation in (20) is over $\boldsymbol{\tau}$ only.
$(\boldsymbol{\varepsilon}, \mathbf{2\ell})$**-Cold:** All but εn variables are $(\varepsilon, 2\ell)$-cold.

A formula Φ is $(\varepsilon, \ell, \lambda)$-*quasirandom* if the properties ℓ-**Local Structure**, **Cycles**, λ-**Core**, λ-**Sticky**, **Gap** and $(\varepsilon, 2\ell)$**-Cold** hold.

▶ **Proposition 13.** *For any $\varepsilon > 0$ there is $\ell > 0$ such that w.h.p. $\hat{\boldsymbol{\Phi}}$ is $(\varepsilon, \ell, 1)$-quasirandom.*

The proof that $\hat{\boldsymbol{\Phi}}$ has the first five properties w.h.p. is based on standard arguments. But the proof of the $(\varepsilon, 2\ell)$**-Cold** property is novel. The argument is intertwined with the study of the Belief Propagation recurrence on the random tree. In particular, that analysis, which we sketch in Section 5, is via a contraction argument that enables a comparison between the result $\nu_T^{\partial\nu}$ for a given bounardy condition and the result for the all-ones boundardy condition (Lemma 20 below). In order to transfer this result from the random tree to the random formula, in which the boundary condition depends on the core, we use a switching argument. The details can be found in the appendix.

Proposition 9 is immediate from Proposition 13 and the following statement.

▶ **Proposition 14.** *For any $\delta > 0$ there exists $\varepsilon > 0$ and $\ell_0(\varepsilon) > 0$ such that for any $\ell > \ell_0(\varepsilon)$ there exists $n_0(\varepsilon, \ell)$ such that for all $n > n_0$ the following is true. If Φ is (ε, ℓ)-quasirandom, then*

$$\frac{1}{n}\sum_{i=1}^{n}\left\langle\left|\nu_{\Delta_\Phi^{2\ell}x_i}^{(2\ell)}(1) - \langle \mathbf{1}\{\boldsymbol{\sigma}(x_i)=1\}|\nabla(\Phi,x_i,\ell,\boldsymbol{\tau})\rangle'\right|\right\rangle'_{\Phi,\beta} < \delta.$$

Proof of Proposition 14

Assume that Φ is (ε, ℓ)-quasirandom and that $n > n_0$ for some large $n_0 = n_0(\varepsilon, \ell)$. In particular, $|\mathrm{Core}_1(\Phi)| \geq (1 - 2^{0.95k})n$. Let $\sigma : V \to \{\pm 1\}$ be an assignment. A set $T \subset \mathrm{Core}_1(\Phi) \setminus S_1(\Phi)$ is σ-*closed* if for any $x \in T$ and all $a \in \partial x$ we have

$$\{y \in \partial a \cap \mathrm{Core}_1(\Phi) \setminus S_1(\Phi) : \sigma(y) = -1\} \subset T. \tag{21}$$

Hence, if $y \in \mathrm{Core}_1(\Phi) \setminus S_1(\Phi)$ is set to -1 and there is a clause a connecting y to some $x \in T$, then y itself must be in T. Moreover, for a clause b we say $T \subset \mathrm{Core}_1(\Phi) \setminus S_1(\Phi)$ is (σ, b)-*closed* if (21) holds for all $x \in T$ and all $a \in \partial x \setminus b$. Additionally, let

$$\partial_{\pm 1,l} x = \{a \in \partial_{\pm 1} x : |\partial_1 a| = l\}$$

be the set of clauses a with a total number of l positive literals where x appears positively/negatively.

▶ **Lemma 15.** *Suppose that Φ is (ε, ℓ)-quasirandom. Then for any σ such that $\mathbf{1} \cdot \sigma \geq (1 - 2^{-k/9})n$ and for any (σ, b)-closed set $T \subset \mathrm{Core}_1(\Phi) \setminus S_1(\Phi)$ the following is true. Let $\tilde{\sigma}(x) = (-1)^{\mathbf{1}\{x \in T\}} \sigma(x)$. Then*

$$E_\Phi(\tilde{\sigma}) \leq E_\Phi(\sigma) - k^{0.98}|T|. \tag{22}$$

Proof. Consider the following process:

- Let $\sigma_0 = \sigma$, $V_0 = T$ and $U_0 = \sigma^{-1}(-1) \setminus V_0$.
- While there is $i_t \in V_t$ such that $E_\Phi((-1)^{\mathbf{1}\{\,\cdot\, = i_t\}} \sigma_t(\,\cdot\,)) \leq E_\Phi(\sigma_t) - k^{0.98}$, pick one such i_t arbitrarily and let $\sigma_{t+1}(\,\cdot\,) = (-1)^{\mathbf{1}\{\,\cdot\, = i_t\}} \sigma_t(\,\cdot\,)$ and $V_{t+1} = V_t \setminus \{i_t\}$.

Clearly,

$$E_\Phi(\sigma_t) \leq E_\Phi(\sigma) - k^{0.98} t. \tag{23}$$

Let τ be the stopping time of this process and assume that $\tau < |T|$, or, in other words, that $V_\tau \neq \emptyset$. We claim that V_τ is a 1-sticky set. Indeed, because T is σ-closed for $i \in V_\tau$ we have

$$\begin{aligned} -k^{0.98} &\leq E_\Phi((-1)^{\mathbf{1}\{\,\cdot\, = i\}} \sigma_t(\,\cdot\,)) - E_\Phi(\sigma_\tau) \\ &\leq \mathbf{1}\{b \in \partial i\} - |\partial_{1,0}(i)| + |\{a \in \partial_{1,0} i, \partial_{-1} a \cap (V_\tau \cup U_0) \neq \emptyset\}| \\ &\quad + |\partial_{-1,0} i| + |\cup_{1 \leq l \leq k} \{a \in \partial_{-1,l} i, \partial_1 a \subset V_\tau \cup U_0)\}|. \end{aligned}$$

Because $i \in \mathrm{Core}_1(\Phi)$ we have $|\partial_{1,0} i| \geq k^{0.99}$, $|\partial_{-1,0} i| \leq 10$, $|\{a \in \partial_{1,0} i, \partial_{-1} a \cap U_0 \neq \emptyset\}| \leq k^{3/4}$ and $|\{a \in \partial_{1,0} i, |\partial_{-1} a \cap U_0| \geq |\partial_{-1} a|/4\}| \leq k^{3/4}$. Therefore, one of the following must hold.

(a) $|\{a \in \partial_{1,0}, \partial_{-1} a \cap V_\tau \neq \emptyset\}| \geq k^{3/4}$,

(b) $|\{a \in \partial_{1,0} i, |\partial_{-1} a \cap V_\tau| \geq |\partial_{-1} a|/4\}| \geq k^{3/4}$.

It follows that the set $V_\tau \subset T \subset \mathrm{Core}_1(\Phi) \setminus S_1(\Phi)$ is 1-sticky.

However, $\mathrm{Core}_1(\Phi) \setminus S_1(\Phi)$ cannot contain a 1-sticky set of size $|V_\tau| \leq |T| \leq 2^{-k/10}$ as this would contradict the maximality of $S_1(\Phi)$. It follows that $\tau = |T|$, and therefore $\sigma_\tau = \tilde{\sigma}$, whence (22) follows using (23). ◀

Lemma 15 is going to be our principal tool to establish Proposition 14. To put it to work, we need the following simple observation that follows from the fact that Φ is d-regular.

▶ **Fact 16.** *For any variable x the following is true. Let $\gamma(x, L)$ be the number of trees with $L \geq 1$ vertices rooted at x that are contained in the factor graph of Φ. Then $\gamma(x, L) \leq L(100dk)^L$.*

Write $T(x, \sigma)$ for the smallest σ-closed set that contains x. If $\sigma(x) = 1$ we let $T(x, \sigma) = \emptyset$. The following lemma shows that $T(x, \sigma)$ is unlikely to be non-empty and very unlikely to be large.

▶ **Lemma 17.** *For all $x \in \mathrm{Core}_1(\Phi) \setminus S_1(\Phi)$ we have*

$$\langle \boldsymbol{\sigma}(x) \rangle' \geq 1 - \exp(-\beta k^{0.97}) \quad \textit{and} \quad \langle \mathbf{1}\{|T(x, \boldsymbol{\sigma})| > \ln \ln n\} \rangle' \leq 1/\ln n.$$

Proof. Let $N = 2^{-k/4}n$. Due to **Gap** we have

$$\langle \mathbf{1}\{\mathbf{1}\cdot\boldsymbol{\sigma} < n - N/2\}\rangle' \leq \exp(-\Omega(n)).$$

Therefore,

$$\langle \mathbf{1}\{|T(x,\boldsymbol{\sigma})| > N\}\rangle' \leq \exp(-\Omega(n)). \tag{24}$$

Hence, let $t \leq N$ and let θ be a tree of order t with root x that is contained in the factor graph of Φ and whose vertices lie in $\mathrm{Core}_1(\Phi)\setminus S_1(\Phi)$. If σ is such that $T(x,\sigma)=\theta$, then Lemma 15 implies that $\tilde\sigma(x) = (-1)^{\mathbf{1}\{x\in T(x,\sigma)\}}\sigma(x)$ satisfies $E_\Phi(\tilde\sigma) \leq E_\Phi(\sigma) - k^{0.98}t$. Consequently,

$$\frac{\langle \mathbf{1}\{\boldsymbol{\sigma}=\sigma\}\rangle'}{\langle \mathbf{1}\{\boldsymbol{\sigma}=\tilde\sigma\}\rangle'} \leq \exp(-\beta k^{0.98}t).$$

Hence, by Fact 16, the union bound and our assumptions on β and d,

$$\frac{\langle \mathbf{1}\{|T(x,\boldsymbol{\sigma})|=t\}\rangle'}{\langle \mathbf{1}\{\boldsymbol{\sigma}(x)=1\}\rangle'} \leq t(100dk)^t\exp(-\beta k^{3/4}t) \leq \exp(-0.99\beta k^{0.98}t). \tag{25}$$

This bound readily implies the second assertion. To obtain the first assertion, we remember (24) and sum (25) over $1 \leq t \leq N$. ◀

Let r be a variable with depth-2ℓ neighborhood T. Guided by (20), we call $\tilde\tau : V \to \{\pm 1\}$ a *good boundary condition* for r if

$$\max\left\{\left|\nu_T^{(2\ell)}(1) - \langle \mathbf{1}\{\boldsymbol{\sigma}(x)=1\}|\nabla(\Phi,x,\ell,\tau)\rangle'\right| : \tau \geq \tilde\tau\right\} \leq \varepsilon. \tag{26}$$

▶ **Lemma 18.** *Let r be a variable for which the following conditions hold.*

1. *r is $(\varepsilon, 2\ell)$-cold.*
2. *r has distance at least $\ln^{1/3} n$ from any cycle of length at most $\sqrt{\ln n}$.*

Let Γ_r be the event that $\boldsymbol{\sigma}$ is a good boundary condition for r. Then $\langle \mathbf{1}\{\boldsymbol{\sigma}\notin\Gamma_r\}\rangle' \leq 2\varepsilon$.

Proof. Let $X = (\partial^{2\ell}r)\cap \mathrm{Core}_1(\Phi)\setminus S_1(\Phi)$. Moreover, let $\mathcal{A}$ be the event that

$$\max_{x\in X}|T(x,\boldsymbol{\sigma})| \leq \ln\ln n \quad \text{and} \quad \boldsymbol{\sigma}\cdot\mathbf{1} \geq (1-2^{-k/4})n.$$

Because $|X| \leq (dk)^\ell < \ln\ln n$ by our assumption that $n > n_0(\varepsilon,\ell)$, Lemma 17 and the fact that Φ is quasirandom imply $\langle \mathbf{1}\{\boldsymbol{\sigma}\in\mathcal{A}\}\rangle' \sim 1$. Furthermore, if $\mathcal{A}$ occurs, then assumption (2) ensures that the subgraph of the factor graph induced on $Y = (\Delta^{2\ell+1}r)\cup\bigcup_{x\in X}T(x,\boldsymbol{\sigma})$ is acyclic.

Now, fix a variable $x\in X$ and $\sigma\in\mathcal{A}$ such that $\sigma(x)=-1$. Let a be the clause that is adjacent to x on its shortest path to r and let $T(x,a,\sigma)$ be the smallest (σ,a)-closed set that contains x. Further, define

$$\tilde\sigma(y) = (-1)^{\mathbf{1}\{y\in T(x,a,\sigma)\}}\sigma(y).$$

Then Lemma 15 shows that $E_\Phi(\tilde\sigma) \leq k^{0.98}|T(x,a,\sigma)|$. Moreover, because the subgraph induced on Y is acyclic we have $\tilde\sigma(x') = \sigma(x')$ for all $x'\in X\setminus\{x\}$. Consequently, by Fact 16 and the union bound,

$$\begin{aligned}\frac{\left\langle \mathbf{1}\{\boldsymbol{\sigma}(x)=-1\}\prod_{y\in X\setminus\{x\}}\mathbf{1}\{\boldsymbol{\sigma}(y)=\sigma(y)\}\mathbf{1}\{\boldsymbol{\sigma}\in\mathcal{A}\}\right\rangle'}{\left\langle \mathbf{1}\{\boldsymbol{\sigma}(x)=1\}\prod_{y\in X\setminus\{x\}}\mathbf{1}\{\boldsymbol{\sigma}(y)=\sigma(y)\}\right\rangle'} &\leq \sum_{t\leq\ln\ln n}\frac{t(100dk)^t}{\exp(\beta k^{0.98}t)} \\ &\leq \exp(-\beta k^{0.98}/2). \end{aligned}\tag{27}$$

Since $\langle \mathbf{1}\{\boldsymbol{\sigma} \in \mathcal{A}\}\rangle' = 1 - o_n(1)$ and because for all $\tau : X \to \{\pm 1\}$ we have

$$\left\langle \prod_{y \in X \setminus \{x\}} \mathbf{1}\{\boldsymbol{\sigma}(y) = \tau(x)\} \right\rangle' \geq \exp(-dk\beta|X|) = \Omega_n(1),$$

(27) implies that for any $\tau : X \to \{\pm 1\}$,

$$\frac{\left\langle \mathbf{1}\{\boldsymbol{\sigma}(x) = -1\} \prod_{y \in X \setminus \{x\}} \mathbf{1}\{\boldsymbol{\sigma}(y) = \tau(y)\} \right\rangle'}{\left\langle \mathbf{1}\{\boldsymbol{\sigma}(x) = 1\} \prod_{y \in X \setminus \{x\}} \mathbf{1}\{\boldsymbol{\sigma}(y) = \tau(y)\} \right\rangle'} \leq \exp(-\beta k^{0.98}/3). \tag{28}$$

Finally, the assertion follows from (28) and the assumption that r is $(\varepsilon, 2\ell)$-cold. ◀

Proof of Proposition 14. The condition **Cycles** ensures that there are at most $o_n(n)$ variables r for which condition (2) from Lemma 18 is violated. Furthermore, due to $(\varepsilon, 2\ell)$-**Cold** all but εn variables r satisfy assumption (1). Therefore, the assertion follows from Lemma 18. ◀

5 Belief Propagation on the infinite tree

Assume that $d \in [d_-(k), d_{k-SAT}]$ and that $\beta \geq \beta_-(k, d)$. Let $c_\beta = 1 - \exp(-\beta)$.

We sketch the analysis of the Belief Propagation messages on the random tree $\boldsymbol{T}_\infty$ to prove Propositions 2 and 11. The key step is the proof of the following statement.

▶ **Lemma 19.** *There exists a number $\ell_0 = \ell_0(d, k, \beta)$ such that for all $\ell \geq \ell_0$ the following is true. Suppose that $\boldsymbol{\partial\nu} : \partial V_{2\ell} \to \{\pm 1\}$ is a random boundary condition, independent of $\boldsymbol{T}_\infty$, such that*
***H:** for any $x \in \partial V_{2\ell}$, $\mathbb{P}\left[(\boldsymbol{\partial\nu})_x(1) \leq 1 - \exp(-k^{0.9}\beta) \,|(\boldsymbol{\partial\nu})_{y \neq x}\right] \leq 2^{-0.9k}$.*
Then

$$\mathbb{P}\left[\|\nu_{\boldsymbol{T}}^{\boldsymbol{\partial\nu}} - \nu_{\boldsymbol{T}}^{(2\ell)}\|_{\mathrm{TV}} \geq 2\ell^{-1}\right] \leq \ell^{-1}.$$

Thus, the condition **H** provides that for any x on the boundary the message $\boldsymbol{\partial\nu}_x(1)$ is likely close to one, even given $\boldsymbol{T}_\infty$ and all the other boundary messages $(\boldsymbol{\partial\nu}_y)_{y \neq x}$. Further, Lemma 19 states that the message at the root given that the boundary condition satisfies **H** is likely within $O_\ell(\ell^{-1})$ of the message obtained from the "all-ones" boundary condition.

We will need a version of Lemma 19 for the random tree $\boldsymbol{T}'_\infty$. Condition **H** becomes **H'**: for any $x \in \partial V'_{2\ell}$, $\mathbb{P}\left[(\boldsymbol{\partial\nu})_x(1) \leq 1 - \exp(-k^{0.9}\beta) \,|(\boldsymbol{\partial\nu})_{y \neq x}\right] \leq 2^{-0.9k}$.

▶ **Lemma 20.** *There is $\ell_0 = \ell_0(d, k, \beta) > 0$ such that for all $\ell \geq \ell_0$ the following is true. Assume that the random boundary condition $\boldsymbol{\partial\nu''}$, independent of $\boldsymbol{T}'$, satisfies **H'**. Moreover, assume that $\boldsymbol{\partial\nu'}$ is a random boundary condition that may depend on $\boldsymbol{T}', \boldsymbol{\partial\nu''}$ such that $\boldsymbol{\partial\nu''}_x(1) \geq \boldsymbol{\partial\nu}'_x(1)$ for all $x \in \partial V'_{2\ell}$. Then for $\ell \geq \ell_0$ we have*

$$\mathbb{P}\left[\|\nu_{\boldsymbol{T}'}^{\boldsymbol{\partial\nu'}} - \nu_{\boldsymbol{T}'}^{(2\ell)}\|_\infty \geq 2\exp(dk\beta)\ell^{-1}\right] \leq 2dk\ell^{-1}.$$

To grasp the assumptions of Proposition 20, we may think of $\boldsymbol{\partial\nu'}$ as obtained from $\boldsymbol{\partial\nu''}$ by allowing an "adversary" to switch some of the -1s of $\boldsymbol{\partial\nu''}$ to $+1$s. The adversary knows both $\boldsymbol{T}'$ and $\boldsymbol{\partial\nu''}$. In the following we tacitly assume that $\ell \geq \ell_0$ for a large constant ℓ_0.

To prove Lemma 19–20 we are going to exhibit a deterministic condition on $(T, \partial\nu)$ that ensures that $\nu_T^{\partial\nu}$ is close to $\nu_T^{(2\ell)}$. For a variable node x we let $\partial_1 x$ be the set of all clauses in which x appears as a positive literal. Similarly, $\partial_{-1}x$ is the set of clauses containing the literal $\neg x$. Conversely, for a clause a we let $\partial_{\pm 1}a$ be the set of all variables that appear in a positively/negatively. Further, we define the *trunk* of T under the boundary condition $\partial\nu$, $\mathrm{Trunk}(T, \partial\nu)$, as the largest subset W of $V_{2\ell}$ such that for any $x \in W$ either

TR0: $x \in \partial V_{2\ell}$ and $\partial\nu_x(1) \geq 1 - \exp(-k^{0.9}\beta)$

or all of the five following conditions hold

TR1: there are at least $\lfloor 2k^{0.9} \rfloor$ clauses $a \in \partial_{\downarrow}x$ such that $\partial_1 a = \{x\}$.

TR2: there are no more than $\lceil \ln k \rceil$ clauses $a \in \partial x$ such that $|\partial_{-1}a| = k$.

TR3: for any $1 \leq l \leq k$ the number of $a \in \partial_{-1}x$ such that $|\partial_1 a| = l$ is bounded by $k^{l+3}/l!$.

TR4: there are no more than $k^{3/4}$ clauses $a \in \partial_1 x$ such that $|\partial_1 a| = 1$ but $\partial a \not\subset W$.

TR5: there are no more than $k^{3/4}$ clauses $a \in \partial_{-1}x$ such that $|\partial_{-1}a| < k$ and $|\partial_1 a \setminus W| \geq |\partial_1 a|/4$.

The trunk is well-defined; for if W, W' are sets that satisfy the above conditions, then so is $W \cup W'$. Somewhat unbotanically, the trunk is non-empty only if it contains some of the leaves. In fact, the construction is monotonous with respect to the boundary condition:

$$\text{if } \partial\nu_x(1) \leq \partial\nu'_x(1) \text{ for all } x \in \partial V_{2\ell}, \text{ then } \mathrm{Trunk}(T, \nu) \subset \mathrm{Trunk}(T, \nu'). \tag{29}$$

For $T \in \mathcal{T}_{2\ell}$ with root r and $x \in \partial V_{2\ell}$, we denote by $[x \to r]$ the shortest path from x to r in T. Moreover,

1. a variable node $x \in V_{2\ell}$ is *cold* if $x \in \mathrm{Trunk}(T, \partial\nu)$,
2. a clause node $a \in F_{2\ell}$ is *cold* if $\partial_1 a \cap \mathrm{Trunk}(T, \partial\nu) \neq \emptyset$,
3. the pair (x, a) with $x \in \partial_{\downarrow}a$ is *cold* if x is cold or a is cold,
4. a path $[x \to r]$ from $x \in \partial V_{2\ell}$ to r is *cold* if it contains at least $\lfloor 0.4\ell \rfloor$ cold pairs (x, a),
5. the pair $(T, \partial\nu) \in \mathcal{T}_{2\ell} \times \mathcal{P}(\{-1, 1\})^{\partial V_{2\ell}}$ is *cold* if all the paths $[x \to r]$ with $x \in \partial V_{2\ell}$ are cold.

The following estimate shows the use of these concepts.

▶ **Lemma 21.** *Assume that $\ell \geq \ell_0(d, k, \beta)$ is sufficiently large. If*

$$(T, \partial\nu) \in \mathcal{T}_{2\ell} \times \mathcal{P}(\{-1, 1\})^{\partial V_{2\ell}}$$

is cold, then

$$\|\nu_{T,\uparrow}^{\partial\nu} - \nu_{T,\uparrow}^{(2\ell)}\|_{\infty} \leq \ell^{-1}.$$

The proof of Lemma 21 is based on a (technically delicate) contraction argument. More precisely, the basic insight is that the Belief Propagation operation is a contraction along cold paths. The proof is based on estimating the derivatives of the formulas (18) and (19). Once contraction is established, convergence of the messages to a unique limit follows just as in the Banach fixed point theorem. Furthermore, a "classical" analysis of the random tree based on Chernoff bounds etc. yields

▶ **Lemma 22.** *Assume that $\ell \geq \ell_0(d, k, \beta)$ is sufficiently large and that $\boldsymbol{\partial\nu}$ satisfies $\boldsymbol{H}$. Then*

$$\mathbb{P}\left[(\boldsymbol{T}, \boldsymbol{\partial\nu}) \textit{ is cold}\right] \geq 1 - \ell^{-1}.$$

Proof of Lemma 19. The assertion follows from the combination of Lemmas 21 and 22 directly. ◀

Proof of Lemma 20. For $h = 1, \ldots, (k-1)d$ consider the sub-trees $\boldsymbol{T}^{(h)}$ of $\boldsymbol{T}'$ pending on the variables at distance exactly two from the root r of $\boldsymbol{T}'$. Then with $\boldsymbol{\partial\nu}^{(h)}$ denoting the boundary condition on $\boldsymbol{T}^{(h)}$ induced by $\boldsymbol{\partial\nu}''$, (29) and Lemma 22 yield

$$\mathbb{P}\left[(\boldsymbol{T}^{(h)}, \boldsymbol{\partial\nu}^{(h)}) \text{ is cold }\right] \geq 1 - (\ell-1)^{-1}.$$

Hence, Lemma 21 yields

$$\mathbb{P}\left[\|\nu_{\boldsymbol{T}^{(h)}}^{\boldsymbol{\partial\nu}^{(h)}} - \nu_{\boldsymbol{T}^{(h)}}^{(2\ell+1)}\|_{\mathrm{TV}} \leq \ell^{-1}\right] \geq 1 - (\ell-1)^{-1}.$$

Therefore, the assertion follows from a coupling argument. ◀

The convergence of the sequence $(\nu_{\boldsymbol{T}_\infty}^{(2\ell)})_{\ell\geq 1}$ follows from Lemma 19 rather directly by induction on ℓ. Similarly, the existence and uniqueness of the distributional fixed point from Proposition 2 follows from Lemma 19, albeit with a bit of work. In fact, it is straightforward to verify that the law of $\nu_{\boldsymbol{T}_\infty}^{\star}(b_{r,\uparrow})$ is a fixed point for Proposition 2. Conversely, any fixed point distribution for Proposition 2 can be "unravelled" to obtain a $\mathcal{P}(\{\pm 1\})$-valued random variable $\lambda_{\boldsymbol{T}_\infty}^{\star}$ such that the original distribution is the law of $\lambda_{\boldsymbol{T}_\infty}^{\star}(b_{r,\uparrow})$ that satisfies condition **H**. The contraction property from Lemma 19 therefore implies that $\lambda_{\boldsymbol{T}_\infty}^{\star}$ coincides with $\nu_{\boldsymbol{T}_\infty}^{\star}$ almost surely. Hence the uniqueness of the distributional fixed point. Similarly, the proof of the $(\varepsilon, 2\ell)$**-Cold** property required in the non-reconstruction argument is based on Lemma 20.

References

1 D. Achlioptas and A. Coja-Oghlan. Algorithmic barriers from phase transitions. In *Proc. 49th FOCS*, pages 793–802, 2008.

2 D. Achlioptas and C. Moore. Random k-sat: two moments suffice to cross a sharp threshold. *SIAM Journal on Computing*, 36:740–762, 2006.

3 D. Achlioptas, A. Naor, and Y. Peres. Rigorous location of phase transitions in hard optimization problems. *Nature*, 435:759–764, 2005.

4 D. Achlioptas and Y. Peres. The threshold for random k-sat is $2^k \ln 2 - o(k)$. *Journal of the AMS*, 17:947–973, 2004.

5 J. Banks and C. Moore. Information-theoretic thresholds for community detection in sparse networks. *Information-theoretic thresholds for community detection in sparse networks*, arXiv:1601.02658, 2016.

6 V. Bapst and A. Coja-Oghlan. Harnessing the bethe free energy. *Harnessing the Bethe free energy*, arXiv:1504.03975, 2015.

7 V. Bapst, A. Coja-Oghlan, S. Hetterich, F. Raßmann, and D. Vilenchik. The condensation phase transition in random graph coloring. *Communications in Mathematical Physics*, 341:543–606, 2016.

8 V. Bapst, A. Coja-Oghlan, and F. Raßmann. A positive temperature phase transition in random hypergraph 2-coloring. *Annals of Applied Probability*, 26:1362?1406, 2016.

9 B. Barak. Structure vs. combinatorics in computational complexity, 2013.

10 M. Bayati, D. Gamarnik, and P. Tetali. Combinatorial approach to the interpolation method and scaling limits in sparse random graphs. *Annals of Probability*, 41:4080–4115, 2013.

11 A. Coja-Oghlan and K. Panagiotou. Going after the k-sat threshold. In *Proc. 45th STOC*, pages 705–714, 2013.

12 A. Coja-Oghlan and K. Panagiotou. The asymptotic k-sat threshold. *Advances in Mathematics*, 288:985–1068, 2016.

13 A. Coja-Oghlan and L. Zdeborová. The condensation transition in random hypergraph 2-coloring. In *Proc. 23rd SODA*, pages 241–250, 2012.

14 P. Contucci, S. Dommers, C. Giardina, and S. Starr. Antiferromagnetic potts model on the Erdős-Rényi random graph. *Communications in Mathematical Physics*, 323:517–554, 2013.

15 A. Dembo, A. Montanari, A. Sly, and N. Sun. The replica symmetric solution for potts models on d-regular graphs. *Comm. Math. Phys.*, 327:551–575, 2014.

16 J. Ding, A. Sly, and N. Sun. Maximum independent sets on random regular graphs. *Maximum independent sets on random regular graphs*, arXiv:1310.4787, 2013.

17 J. Ding, A. Sly, and N. Sun. Satisfiability threshold for random regular nae-sat. In *Proc. 46th STOC*, pages 814–822, 2014.

18 J. Ding, A. Sly, and N. Sun. Proof of the satisfiability conjecture for large k. In *Proc. 47th STOC*, pages 59–68, 2015.

19 D. Gamarnik, T. Nowicki, and G. Swirszcz. Maximum weight independent sets and matchings in sparse random graphs: Exact results using the local weak convergence method. *Random Structures and Algorithms*, 28:76–106, 2006.

20 D. Gamarnik and M. Sudan. Limits of local algorithms over sparse random graphs. In *Proc. 5th ITCS*, pages 369–376, 2014.

21 D. Gamarnik and M. Sudan. Performance of the survey propagation-guided decimation algorithm for the random nae-k-sat problem. *Performance of the Survey Propagation-guided decimation algorithm for the random NAE-K-SAT problem*, arXiv:1402.0052, 2014.

22 F. Krzakala, A. Montanari, F. Ricci-Tersenghi, G. Semerjian, and L. Zdeborova. Gibbs states and and the set of solutions of random constraint satisfaction problems. *Proc. National Academy of Sciences*, 104:10318–10323, 2007.

23 M. Mézard and A. Montanari. *Information, physics and computation.* Oxford University Press, 2009.

24 M. Mézard, G. Parisi, and M. Virasoro. *Spin Glass Theory and Beyond.* World Scientific, 1987.

25 M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297:812–815, 2002.

26 E. Mossel, J. Neeman, and A. Sly. Reconstruction and estimation in the planted partition model. *Probability Theory and Related Fields*, pages 1–31, 2014.

27 D. Panchenko. On the replica symmetric solution of the k-sat model. *Electron. J. Probab.*, 19:1–17, 2014.

28 V. Rathi, E. Aurell, L. K. Rasmussen, and M. Skoglund. Bounds on threshold of regular random k-sat. In *Proc. 12th SAT*, pages 264–277, 2010.

29 F. Ricci-Tersenghi and G. Semerjian. On the cavity method for decimated random constraint satisfaction problems and the analysis of belief propagation guided decimation algorithms. *J. Stat. Mech.*, 9001, 2009.

30 T. Richardson and R. Urbanke. *Modern coding theory.* Cambridge University Press, 2008.

31 M. Talagrand. The high temperature case for the random k-sat problem. *Probab. Theory Related Fields*, 119:187–212, 2001.

32 L. Zdeborová and F. Krzakala. Statistical physics of inference: Thresholds and algorithms. *Statistical physics of inference: thresholds and algorithms*, arXiv:1511.02476, 2015.

On Higher-Order Fourier Analysis over Non-Prime Fields

Arnab Bhattacharyya[1], Abhishek Bhowmick[2], and Chetan Gupta[3]

1 **Department of Computer Science & Automation, Indian Institute of Science, India**
`arnabb@csa.iisc.ernet.in`

2 **Department of Computer Science, The University of Texas at Austin, USA**
`bhowmick@cs.utexas.edu`

3 **Department of Computer Science & Automation, Indian Institute of Science, India**
`chetan.gupta@csa.iisc.ernet.in`

Abstract

The celebrated Weil bound for character sums says that for any low-degree polynomial P and any additive character χ, either $\chi(P)$ is a constant function or it is distributed close to uniform. The goal of higher-order Fourier analysis is to understand the connection between the algebraic and analytic properties of polynomials (and functions, generally) at a more detailed level. For instance, what is the tradeoff between the equidistribution of $\chi(P)$ and its "structure"?

Previously, most of the work in this area was over fields of prime order. We extend the tools of higher-order Fourier analysis to analyze functions over general finite fields. Let $\mathbb{K}$ be a field extension of a prime finite field $\mathbb{F}_p$. Our technical results are:

1. If $P : \mathbb{K}^n \to \mathbb{K}$ is a polynomial of degree $\leqslant d$, and $\mathbb{E}[\chi(P(x))] > |\mathbb{K}|^{-s}$ for some $s > 0$ and non-trivial additive character χ, then P is a function of $O_{d,s}(1)$ many *non-classical polynomials* of weight degree $< d$. The definition of non-classical polynomials over non-prime fields is one of the contributions of this work.
2. Suppose $\mathbb{K}$ and $\mathbb{F}$ are of bounded order, and let H be an affine subspace of $\mathbb{K}^n$. Then, if $P : \mathbb{K}^n \to \mathbb{K}$ is a polynomial of degree d that is sufficiently regular, then $(P(x) : x \in H)$ is distributed almost as uniformly as possible subject to constraints imposed by the degree of P. Such a theorem was previously known for H an affine subspace over a prime field.

The tools of higher-order Fourier analysis have found use in different areas of computer science, including list decoding, algorithmic decomposition and testing. Using our new results, we revisit some of these areas.

(i) For any fixed finite field $\mathbb{K}$, we show that the list decoding radius of the generalized Reed Muller code over $\mathbb{K}$ equals the minimum distance of the code.
(ii) For any fixed finite field $\mathbb{K}$, we give a polynomial time algorithm to decide whether a given polynomial $P : \mathbb{K}^n \to \mathbb{K}$ can be decomposed as a particular composition of lesser degree polynomials.
(iii) For any fixed finite field $\mathbb{K}$, we prove that all locally characterized affine-invariant properties of functions $f : \mathbb{K}^n \to \mathbb{K}$ are testable with one-sided error.

1998 ACM Subject Classification G.2 Discrete Mathematics, D.2.5 Testing and Debugging

Keywords and phrases finite fields, higher order fourier analysis, coding theory, property testing

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.23

1 Introduction

In this work, we provide new results about polynomials over finite fields, relating their algebraic structure to the distribution of their output. We then apply them to improve previous work on list-decoding bounds for the Reed-Muller code, testability of affine-invariant properties, and algorithms for polynomial decomposition.

1.1 Structure versus Randomness for Polynomials over Finite Fields

In many areas of mathematics, there is a remarkable phenomenon where natural objects are either close to random or have a high degree of structure. A prime example of this is the Weil bound for character sums [49], a deep result in algebraic geometry.

Let $\mathbb{F}$ be a finite field of prime order p, and let $\mathbb{K}$ be a finite field extension of $\mathbb{F}$. Let $P : \mathbb{K} \to \mathbb{K}$ be a univariate polynomial of degree $\leqslant |\mathbb{K}|^{1/2-\delta}$ for some $\delta > 0$. If we let $\chi : \mathbb{K} \to \mathbb{C}$ denote a non-trivial additive character, then according to Weil's bound, either $\chi(P(x))$ is constant or else, $\chi(P(x))$ is distributed close to uniform in the sense that $|\,\mathbb{E}[\chi(P(x))]| \leqslant |\mathbb{K}|^{-\delta}$. Deligne later [17] proved the same statement for multivariate polynomials $P : \mathbb{K}^n \to \mathbb{K}$.

Higher-order Fourier analysis [46] is a recent generalization of some aspects of Fourier analysis. Over finite fields, one of the main components of the theory is a detailed study of the interplay between the algebraic structure and analytic properties of polynomials. Consider the case when $\mathbb{K} = \mathbb{F}_p$ is a prime field and p is small, for example 2. Weil's bound does not apply for $d > \sqrt{p}$. However, in the context of higher-order Fourier analysis, Kaufmann and Lovett [37] (extending previous work by Green and Tao [30]) showed that if $P : \mathbb{F}_p^n \to \mathbb{F}_p$ is a polynomial of degree d, then for any non-trivial additive character χ, either $|\,\mathbb{E}[\chi(P(x))]| < \varepsilon$ or else, P is a function of a $O_{\varepsilon,d,p}(1)$ polynomials of strictly lesser degree. The regime here is different from that of Weil's bound (large n versus large $|\mathbb{K}|$) but the result is similar in spirit.

These recent developments have spurred a deeper look at the dichotomy between randomness and structure in polynomials over finite fields. For instance, instead of using the bias, $|\,\mathbb{E}[\chi(P(x))]|$, as a measure of randomness, one can look at how well P is equidistributed on affine subspaces. This gives rise to the *Gowers norm* [28, 29], a central notion in higher-order Fourier analysis. Low Gowers norm is a much stronger notion of pseudorandomness than low bias. Green and Tao [30] showed the *Gowers inverse theorem*, which states that if $P : \mathbb{F}_p^n \to \mathbb{F}_p$ is a polynomial of degree $d < p$, then either its Gowers norm of order d is smaller than ε or it is a function of $O_{\varepsilon,d,p}(1)$ polynomials of degree $< d$. When $d \geqslant p$, this dichotomy fails to be true [40, 30]; Tao and Ziegler [47] showed that if the Gowers norm is not small, then the polynomial is a function of a bounded number of *non-classical polynomials*, functions mapping to the torus that locally look like low-degree polynomials.

All of these results focused on finite fields of prime order. In general fields, the situation is more complicated for several reasons. Firstly, the additive characters over $\mathbb{F}$ are simply exponential functions $\chi_a(x) = e^{2\pi i a x/p}$ for $a \neq 0$ which are bijective, while over general fields $\mathbb{K}$, the additive characters are $\chi_a(x) = e^{2\pi i \mathrm{Tr}(ax)/p}$ for $a \neq 0$ where the trace $\mathrm{Tr} : \mathbb{K} \to \mathbb{F}$ is a linear map with a possibly large kernel. Secondly, if $\mathbb{K}$ is of dimension at least 2 over $\mathbb{F}$, polynomials like x^p have degree p but vanish after taking only 2 derivatives. Define the *weight degree* of a polynomial $P : \mathbb{K}^n \to \mathbb{K}$ to be the minimum number of derivatives before P becomes identically a constant. If a variable has individual degree more than p in a monomial in P, then the weight degree of P may not equal the total degree. Thirdly, while $\mathbb{K}$ can often be profitably viewed as simply a vector space over $\mathbb{F}$, affine subspaces in $\mathbb{K}^n$ are not necessarily vector spaces over $\mathbb{F}$.

In this work, we prove the first structure-versus-randomness dichotomy result for polynomials over general fields in the higher-order Fourier analysis setting. We show that if $P : \mathbb{K}^n \to \mathbb{K}$ is a polynomial of total degree d, then either its Gowers norm of order d is less than $|\mathbb{K}|^{-s}$ (and hence, so is the bias of any additive character of P), or else, P is a function of $O_{d,s}(1)$ many *non-classical polynomials* over $\mathbb{K}^n$ of weight degree less than d. Our definition of non-classical polynomials uses the multiplicative structure of $\mathbb{K}$. Also, note that unlike the results quoted above, our result is non-trivial when $|\mathbb{K}|$ is not constant. For constant d, our result continues to give information about the structure of the polynomial even when its bias is less than $|\mathbb{K}|^{-1/2}$, the limit of Weil's bound. To make our bounds hold in this regime, we use some results recently given by Bhowmick and Lovett [13] where they mainly[1] studied higher-order Fourier analysis over growing prime fields.

Our next dichotomy result holds for constant sized $\mathbb{K}$. Let c be a positive integer bound, and let $P : \mathbb{K}^n \to \mathbb{K}$ be a polynomial of weight degree d. Then, either for some $\alpha \in \mathbb{K}$, $\mathrm{Tr}(\alpha P)$ is expressible in terms of polynomials of lower degree, or the distribution of P on random affine subspaces of $\mathbb{K}^n$ of dimension c is as uniform as possible subject to the constraints imposed by the weight degree of P. For instance, if the weight degree of P is 1, then $P(x+y+z) - P(x+y) - P(x+z) + P(x) = 0$ for all $x, y, z \in \mathbb{K}^n$ and hence, this constraint must be satisfied by the evaluations of P on any affine subspace. The second possibility in the dichotomy is that modulo such constraints, the value of P is almost unconstrained on subspaces of dimension c.

1.2 Applications

In this section, we describe three different problems involving a finite field $\mathbb{K}$, which previously had been solved only when $|\mathbb{K}|$ was prime but which we can now solve for arbitrary finite $\mathbb{K}$.

Throughout, let $\mathbb{F}$ be a fixed prime order field, and let $\mathbb{K}$ be a finite field that extends $\mathbb{F}$. Let $q = |\mathbb{K}|$, $p = |\mathbb{F}|$ and $q = p^r$ for $r > 0$.

1.2.1 List-decoding Reed-Muller codes

The notion of *list decoding* was introduced by Elias [18] and Wozencraft [50] to decode *error correcting codes* beyond half the minimum distance. The goal of a list decoding algorithm is to produce all the codewords within a specified distance from the received word. At the same time one has to find the right radius for which the number of such codewords is small, otherwise there is no hope for the algorithm to be efficient. After the seminal results of Goldreich and Levin [20] and Sudan [43] which gave list decoding algorithms for the Hadamard code and the Reed-Solomon code respectively, there has been tremendous progress in designing list decodable codes. See the survey by Guruswami [34, 33] and Sudan [44].

Reed-Muller codes (RM codes) were discovered by Muller in 1954. Let $d \in \mathbb{N}$. The RM code $\mathrm{RM}_{\mathbb{K}}(n, d)$ is defined as follows. The message space consists of degree $\leqslant d$ polynomials in n variables over $\mathbb{K}$ and the codewords are evaluation of these polynomials on $\mathbb{K}^n$. Let $\delta_q(d)$ denote the normalized distance of $\mathrm{RM}_{\mathbb{K}}(n, d)$. Let $d = a(q-1) + b$ where $0 \leqslant b < q-1$. We have

$$\delta_{\mathbb{K}}(d) = \frac{1}{q^a}\left(1 - \frac{b}{q}\right).$$

[1] As we discuss later, they also study non-prime fields in Section 4.9 but they restrict to the case $d < p$ whereas we focus on small p.

RM codes are one of the most well studied error correcting codes. Many applications in computer science involve low degree polynomials over small fields, namely RM codes. Given a received word $g : \mathbb{K}^n \to \mathbb{K}$ the objective is to output the list of codewords (e.g. low-degree polynomials) that lie within some distance of g. Typically we will be interested in regimes where list size is either independent of n or polynomial in the block length q^n.

Let $\mathcal{P}_d(\mathbb{K}^n)$ denote the class of degree $\leqslant d$ polynomials $f : \mathbb{F}^n \to \mathbb{F}$. Let dist denote the normalized Hamming distance. For $\mathrm{RM}_{\mathbb{K}}(n, d)$, $\eta > 0$, let

$$\ell_{\mathbb{F}}(n, d, \eta) := \max_{g:\mathbb{F}^n \to \mathbb{F}} |\{f \in \mathcal{P}_d(\mathbb{F}^n) : \mathrm{dist}(f, g) \leqslant \eta\}| .$$

Let $\mathrm{LDR}_{\mathbb{K}}(n, d)$ (short for *list decoding radius*) be the maximum ρ for which $\ell_{\mathbb{K}}(n, d, \rho - \varepsilon)$ is upper bounded by a constant depending only on $\varepsilon, |\mathbb{K}|, d$ for all $\varepsilon > 0$.

It is easy to see that $\mathrm{LDR}_{\mathbb{K}}(n, d) \leqslant \delta_{\mathbb{K}}(d)$. The difficulty lies in proving a matching lower bound. We review some previous work next. The first breakthrough result was the celebrated work of Goldreich and Levin [20] who showed that in the setting of $d = 1$ over $\mathbb{F}_2$ (Hadamard Codes) $\mathrm{LDR}_{\mathbb{F}_2}(n, 1) = \delta_{\mathbb{F}_2}(1) = 1/2$. Later, Goldreich, Rubinfield and Sudan [21] generalized the field to obtain $\mathrm{LDR}_{\mathbb{K}}(n, 1) = \delta_{\mathbb{K}}(1) = 1 - 1/|\mathbb{K}|$. In the setting of $d < |\mathbb{K}|$, Sudan, Trevisan and Vadhan [45] showed that $\mathrm{LDR}_{\mathbb{K}}(n, d) \geqslant 1 - \sqrt{2d/|\mathbb{K}|}$ improving previous work by Arora and Sudan [2], Goldreich *et al* [21] and Pellikaan and Wu [41]. Note that this falls short of the upper bound which is $\delta_{\mathbb{K}}(d)$.

In 2008, Gopalan, Klivans and Zuckerman [26] showed that $\mathrm{LDR}_{\mathbb{F}_2}(n, d) = \delta_{\mathbb{F}_2}(d)$.They posed the following conjecture.

▶ **Conjecture 1** ([26]). *For fixed d and finite field $\mathbb{K}$, $\mathrm{LDR}_{\mathbb{K}}(n, d) = \delta_{\mathbb{K}}(d)$.*

It is believed [26, 25] that the hardest case is the setting of small d. An important step in this direction was taken in [25] that considered quadratic polynomials and showed that $\mathrm{LDR}_{\mathbb{K}}(n, 2) = \delta_{\mathbb{K}}(2)$ for all fields $\mathbb{K}$ and thus proved the conjecture for $d = 2$. Recently, Bhowmick and Lovett [14] resolved the conjecture for prime $\mathbb{K}$.

Our main result for list decoding is a resolution of Conjecture 1.

▶ **Theorem 2.** *Let $\mathbb{K}$ be a finite field. Let $\varepsilon > 0$ and $d, n \in \mathbb{N}$. Then,*

$$\ell_{\mathbb{K}}(d, n, \delta_{\mathbb{K}}(d) - \varepsilon) \leqslant c_{|\mathbb{K}|,d,\varepsilon}.$$

Thus,

$$\mathrm{LDR}_{\mathbb{K}}(n, d) = \delta_{\mathbb{K}}(d).$$

▶ Remark (Algorithmic Implications). Using the blackbox reduction of algorithmic list decoding to combinatorial list decoding in [26] along with Theorem 18, for fixed finite fields, d and $\varepsilon > 0$, we now have list decoding algorithms in both the global setting (running time polynomial in $|\mathbb{K}|^n$) and the local setting (running time polynomial in n^d).

▶ Remark. Note that the bound on the list size in Theorem 2 depends on $|\mathbb{K}|$. The recent work of Bhowmick and Lovett [13] shows that this is not necessary when $d < p$. It is an open question to show this for general fields. Our dichotomy result for polynomials on general fields can't be used in their proof because it only holds for polynomials mapping to $\mathbb{K}$, not for the more general non-classical polynomials.

1.2.2 Algorithmic polynomial decomposition

Consider the following family of properties of functions over a finite field $\mathbb{K}$.

▶ **Definition 3.** Given a positive integer k, a vector of positive integers $\mathbf{\Delta} = (\Delta_1, \Delta_2, \dots, \Delta_k)$ and a function $\Gamma : \mathbb{K}^k \to \mathbb{K}$, we say that a function $P : \mathbb{K}^n \to \mathbb{K}$ is $(k, \mathbf{\Delta}, \Gamma)$-*structured* if there exist polynomials $P_1, P_2, \dots, P_k : \mathbb{K}^n \to \mathbb{K}$ with each $\deg(P_i) \leqslant \Delta_i$ such that for all $x \in \mathbb{K}^n$,

$$P(x) = \Gamma(P_1(x), P_2(x), \dots, P_k(x)).$$

The polynomials $P_1, \dots, P_k$ are said to form a $(k, \mathbf{\Delta}, \Gamma)$-*decomposition.*

For instance, an n-variate polynomial over the field $\mathbb{K}$ of total degree d factors nontrivially exactly when it is $(2, (d-1, d-1), \mathsf{prod})$-structured where $\mathsf{prod}(a, b) = a \cdot b$. We shall use the term *degree-structural property* to refer to a property from the family of $(k, \mathbf{\Delta}, \Gamma)$-structured properties.

The problem here is, for arbitrary fixed $k, \mathbb{K}, \mathbf{\Delta}, \Gamma$, given a polynomial, decide efficiently if it is degree structural and if yes, output the decomposition. An efficient algorithm for the above would imply a (deterministic) $\text{poly}(n)$-time algorithm for factoring an n-variate polynomial of degree d over $\mathbb{K}$. Also, it implies a polynomial time algorithm for deciding whether a d-dimensional tensor over $\mathbb{K}$ has rank at most r. Also, it would give polynomial time algorithms for a wide range of problems not known to have non-trivial solutions previously, such as whether a polynomial of degree d can be expressed as $P_1 \cdot P_2 + P_3 \cdot P_4$ where each P_1, P_2, P_3, P_4 are of degree $d - 1$ or less.

This problem was solved for prime $\mathbb{K} = \mathbb{F}$, satisfying $d < p$ by Bhattacharyya [6] and later for all d and prime $\mathbb{F}$ by Bhattacharyya, Hatami and Tulsiani [12]. Our main result in this line of work establishes this for all fixed finite fields.

▶ **Theorem 4.** *For every finite field $\mathbb{K}$, positive integers k and d, every vector of positive integers $\mathbf{\Delta} = (\Delta_1, \Delta_2, \dots, \Delta_k)$ and every function $\Gamma : \mathbb{K}^k \to \mathbb{K}$, there is a deterministic algorithm $\mathcal{A}_{\mathbb{K},d,k,\mathbf{\Delta},\Gamma}$ that takes as input a polynomial $P : \mathbb{K}^n \to \mathbb{K}$ of degree d that runs in time polynomial in n, and outputs a $(k, \mathbf{\Delta}, \Gamma)$-decomposition of P if one exists while otherwise returning* NO*.*

1.2.3 Testing affine-invariant properties

The goal of property testing, as initiated by [15, 4] and defined formally by [42, 22], is to devise algorithms that query their input a very small number of times while correctly deciding whether the input satisfies a given property or is "far" from satisfying it. A property is called *testable* if the query complexity can be made independent of the size of the input.

More precisely, we use the following definitions. Let $[R]$ denote the set $\{1, \dots, R\}$. Given a property $\mathcal{P}$ of functions in $\{\mathbb{K}^n \to [R] \mid n \in \mathbb{Z}_{\geqslant 0}\}$, we say that $f : \mathbb{K}^n \to [R]$ is ε-*far* from $\mathcal{P}$ if

$$\min_{g \in \mathcal{P}} \mathbf{Pr}_{x \in \mathbb{K}^n}[f(x) \neq g(x)] > \varepsilon,$$

and we say that it is ε-*close* otherwise.

▶ **Definition 5** (Testability)**.** A property $\mathcal{P}$ is said to be *testable* (with one-sided error) if there are functions $q : (0, 1) \to \mathbb{Z}_{>0}$, $\delta : (0, 1) \to (0, 1)$, and an algorithm T that, given as input a parameter $\varepsilon > 0$ and oracle access to a function $f : \mathbb{K}^n \to [R]$, makes at most $q(\varepsilon)$

queries to the oracle for f, always accepts if $f \in \mathcal{P}$ and rejects with probability at least $\delta(\varepsilon)$ if f is ε-far from $\mathcal{P}$. If, furthermore, q is a constant function, then $\mathcal{P}$ is said to be *proximity-obliviously testable (PO testable).*

The term proximity-oblivious testing is coined by Goldreich and Ron in [24]. As an example of a testable (in fact, PO testable) property, let us recall the famous result by Blum, Luby and Rubinfeld [15] which initiated this line of research. They showed that linearity of a function $f : \mathbb{K}^n \to \mathbb{K}$ is testable by a test which makes 3 queries. This test accepts if f is linear and rejects with probability $\Omega(\varepsilon)$ if f is ε-far from linear.

Linearity, in addition to being testable, is also an example of a *linear-invariant* property. We say that a property $\mathcal{P} \subseteq \{\mathbb{K}^n \to [R]\}$ is linear-invariant if it is the case that for any $f \in \mathcal{P}$ and for any $\mathbb{K}$-linear transformation $L : \mathbb{K}^n \to \mathbb{K}^n$, it holds that $f \circ L \in \mathcal{P}$. Similarly, an *affine-invariant* property is closed under composition with affine transformations $A : \mathbb{K}^n \to \mathbb{K}^n$ (an affine transformation A is of the form $L + c$ where L is $\mathbb{K}$-linear and $c \in \mathbb{K}$). The property of a function $f : \mathbb{K}^n \to \mathbb{K}$ being affine is testable by a simple reduction to [15], and is itself affine-invariant. Other well-studied examples of affine-invariant (and hence, linear-invariant) properties include Reed-Muller codes [4, 3, 19, 42, 1] and Fourier sparsity [27]. In fact, affine invariance seems to be a common feature of most interesting properties that one would classify as "algebraic". Kaufman and Sudan in [39] made explicit note of this phenomenon and initiated a general study of the testability of affine-invariant properties (see also [23]).

Our main theorem for testing is a very general positive result:

▶ **Theorem 6** (Main testing result). *Let $\mathcal{P} \subseteq \{\mathbb{K}^n \to [R]\}$ be an affine-invariant property that is t, w-lightly locally characterized, where t, R, w, and* $\mathrm{char}(\mathbb{K})$ *are fixed positive integers. Then, $\mathcal{P}$ is PO testable with t queries.*

We are yet to define several terms in the above claim, but as we will see, the weight restriction is trivial when the field size is bounded. This yields the following characterization.

▶ **Theorem 7** (Testing result for fixed fields). *Let $\mathcal{P} \subseteq \{\mathbb{K}^n \to [R]\}$ be an affine-invariant property, where $R \in \mathbb{Z}^+$ and field $\mathbb{K}$ are fixed. Then, $\mathcal{P}$ is PO testable with t queries if and only if $\mathcal{P}$ is t-locally characterized.*

Previously, [9] (building on [8, 11, 10]) proved Theorem 6 in the case that $\mathbb{K}$ is of fixed prime order using higher-order Fourier analytic techniques. We note that other recent results on 2-sided testability of affine-invariant properties over fixed prime-order fields [35, 51] can also be similarly extended to non-prime fields but we omit their description here.

1.2.3.1 Local Characterizations

For a PO testable property $\mathcal{P} \subset \{\mathbb{K}^n \to [R]\}$ of query complexity t, if a function $f : \mathbb{K}^n \to [R]$ does not satisfy $\mathcal{P}$, then by Definition 5, the tester rejects f with positive probability. Since the test always accepts functions with the property, there must be t points $a_1, \ldots, a_t \in \mathbb{K}^n$ that form a witness for non-membership in $\mathcal{P}$. These are the queries that cause the tester to reject. Thus, denoting $\sigma = (f(a_1), \ldots, f(a_t)) \in [R]^t$, we say that $\mathcal{C} = (a_1, a_2, \ldots, a_t; \sigma)$ forms a *t-local constraint* for $\mathcal{P}$. This means that whenever the constraint is violated by a function g, i.e., $(g(a_1), \ldots, g(a_t)) = \sigma$, we know that g is not in $\mathcal{P}$. A property $\mathcal{P}$ is *t-locally characterized* if there exists a collection of t-local constraints $\mathcal{C}_1, \ldots, \mathcal{C}_m$ such that $g \in \mathcal{P}$ if and only if none of the constraints $\mathcal{C}_1, \ldots, \mathcal{C}_m$ are violated. It follows from the above discussion that if $\mathcal{P}$ is PO testable with q queries, then $\mathcal{P}$ is t-locally characterized.

For an affine-invariant property, constraints can be defined in terms of affine forms, since the affine orbit of a constraint is also a constraint. So, we can describe each t-local constraint

$\mathcal{C}$ as $(A_1, \ldots, A_t; \sigma)$, where for every $i \in [t]$, $A_i(X_1, \ldots, X_t) = X_1 + \sum_{j=2}^{t} c_{i,j} X_j$ for some $c_{i,j} \in \mathbb{K}$ is an affine form over $\mathbb{K}$. We define the *weight* wt of an element $c \in \mathbb{K}$ as $\sum_{k=1}^{r} |c_k|$, where c is viewed as an r-dimensional vector $(c_1, \ldots, c_r)$ with each c_i in the base prime field[2] $\mathbb{F}$ with respect to a fixed arbitrary basis. The *weight of an affine form* A_i to be $\sum_{j=2}^{m} \mathsf{wt}(c_{i,j})$ for $c_{i,j}$ as above. A constraint is said to be of weight w if all its affine forms are of weight at most w, and a property $\mathcal{P}$ is said to be t, w-lightly locally characterized if there exist t-local constraints $\mathcal{C}_1, \ldots, \mathcal{C}_m$, each of weight at most w that characterize $\mathcal{P}$.

Theorem 6 asserts that if $\mathcal{P}$ has a light local characterization, then it is testable. There can exist many local characterizations of a property, and for the theorem to apply, it is only necessary that one such characterization be of bounded weight. Moreover, we can choose the basis with which to describe $\mathbb{K}$ over $\mathbb{F}$. On the other hand, some restriction in addition to local characterization is needed, as Ben-Sasson et al. [5] show that there exist affine-invariant locally characterized properties of functions $f : \mathbb{F}_{2^n} \to \mathbb{F}_2$ that require super-constant query complexity to test.

Another interesting observation is that if a property has a local characterization of bounded weight, then it has a local *single orbit characterization*, in the language of [39]. For linear[3] affine-invariant properties, [39] shows that any local single orbit characterized property is testable. Hence, our result is weaker than [39] in this aspect, though our Theorem 6 allows non-linear properties. It is an interesting open question as to whether dual-BCH codes and, more generally, sparse affine-invariant codes that were shown to be locally single orbit characterized in [36] and [31] respectively also have local characterizations of bounded weight. It is also an open problem to describe a testable property $\mathcal{P} \subseteq \{\mathbb{F}_{2^n} \to \mathbb{F}_2\}$ that does not have a local characterization of bounded weight.

1.3 Parallel Work

Subsequent to our first public version of this work [7], Bhowmick and Lovett [13] proved Theorem 2 for all finite fields, even when the field size is growing with n, but assuming that the field characteristic is larger than the order of the Reed-Muller code. They focus more on handling growing field size instead of arbitrary field characteristic, so their techniques are substantially different.

1.4 Our Techniques

1.4.1 New Ingredients

Our starting point is the observation that $\mathbb{K}$ is an r-dimensional vector space over $\mathbb{F}$. Thus, we can view a function $Q : \mathbb{K}^n \to \mathbb{K}$ as determined by a collection of functions $P_1, \ldots, P_r : \mathbb{K}^n \to \mathbb{F}$ where $\mathbb{K}^n$ is viewed as $\mathbb{F}^{rn}$. However, it is incorrect to suppose $P_1, \ldots, P_r$ are independent as they are generated by the same polynomial over $\mathbb{K}$.

Indeed, in our first dichotomy theorem, we want to deduce structural information about P just from the fact that P_1 is biased. Although we can't directly prove that biased P_1 implies biased P_i for all $i \in [r]$, we show that biased DP_1 implies biased DP_i for all $i \in [r]$, where for a polynomial Q of degree d, $DQ(h_1, \ldots, h_d)$ is the iterated derivative of Q in directions $h_1, \ldots, h_d$. Because d is the total degree of the polynomial, the iterated derivative

[2] If $x \in \mathbb{F}$, $|x|$ is the obvious element of $\{0, 1, \ldots, |\mathbb{F}| - 1\}$.
[3] These are properties of functions $f : \mathbb{K}^n \to \mathbb{F}$, where $\mathbb{F}$ is a subfield of $\mathbb{K}$, for which $f, g \in \mathcal{P}$ implies $\alpha f + \beta g \in \mathcal{P}$ for any $\alpha, \beta \in \mathbb{F}$.

is multilinear in $h_1, \ldots, h_d$. The multilinearity allows us to relate the structure of DP_1 to the rest of the DP_is. The same strategy was used in [13]. We then follow the steps of [48] to integrate each of the DP_i's. We show that all of them are functions of the same collection of *non-classical polynomials*. Here, a non-classical polynomial $Q(x_1, \ldots, x_n)$ is determined by a set of monomials $ax_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$ where a and the multiplication is in $\mathbb{K}$; the evaluation of each monomial is then mapped to $(\mathbb{Z}/p^{k+1}\mathbb{Z})^r$ for some integer $k \geqslant 0$ and the final output is an integer linear combination of them. Note that unlike our definition, in the definition of non-classical polynomials over $\mathbb{F}^n$ by [48], the multiplicative structure of the field is never used. Also, our non-classical polynomials are a strict subset of the functions $P : \mathbb{K}^n \to \mathbb{T}^r$ which identically vanish after $d + 1$ derivatives. In fact, if we had used the latter notion as defining non-classical polynomials, our theorem would have been quite straightforward.

Over constant-sized fields $\mathbb{K}$, it is more economical to write out P as determined by $P_1, \ldots, P_r$ and treat them as independent. Then, we have reduced the problem to studying polynomials mapping to $\mathbb{F}$. However, even in this setting, we cannot totally ignore the multiplicative structure of $\mathbb{K}$. To see why, recall the question of testing affine-invariant properties. When $\mathbb{K}$ is of bounded order, we can view any one-sided test as examining the restriction of the input function on a random K-dimensional affine subspace of $\mathbb{K}^n$, for some constant integer K. In other words, the test will evaluate the input function at elements of the set $H = \{x + \sum_{i=1}^{K} a_i y_i : a_1, \ldots, a_K \in \mathbb{K}\}$ for some $x, y_1, \ldots, y_K \in \mathbb{K}$. Clearly, H is not an affine subspace of $\mathbb{F}^{rn}$ because of $\mathbb{K}$'s multiplicative structure. An important component of the higher-order Fourier analytic approach is to show that any "sufficiently pseudorandom" collection of polynomials is equidistributed on H, and the proof of this fact in [9] crucially uses that H is a subspace of a vector space over a prime field. In our work, we show a strong equidistribution theorem (Theorem 36) that holds when H is an affine subspace of $\mathbb{K}^n$.

A different place where the multiplicative structure of $\mathbb{K}$ rears its head is a key *Degree Preserving Lemma* of [9]. Informally, if $P_1, \ldots, P_C$ form a "sufficiently pseudorandom" collection of polynomials and $F(x) = \Gamma(P_1(x), \ldots, P_C(x))$ is a polynomial of degree d where Γ is an arbitrary composition function, then for any other collection of polynomials $Q_1, \ldots, Q_C$ where $\deg(Q_i) \leqslant \deg(P_i)$ for every i, $G(x) = \Gamma(Q_1(x), \ldots, Q_C(x))$ also has degree $\leqslant d$. The lemma is crucially used for the analysis of the Reed-Muller list decoding bound in [14] and the polynomial decomposition algorithm in [6, 12]. Its proof goes via showing that if all $(d+1)$ iterated derivatives of $F : \mathbb{K}^n \to \mathbb{K}$ vanish, then so must all $(d+1)$ iterated derivatives of $G : \mathbb{K}^n \to \mathbb{K}$. However, for $\mathbb{K}$ that is of size p^2 or more, this only implies a bound on the weight degree of G, not on its degree.

We resolve this issue by giving a different and more transparent proof of the Degree Preserving Lemma, which actually holds in a much more general setting. Using the above notation, we prove that if $F : \mathbb{K}^n \to \mathbb{K}$ satisfies some locally characterized property $\mathcal{P}$, then $G : \mathbb{K}^n \to \mathbb{K}$ does also. Since due to a work of Kaufman and Ron [38], we know that degree is locally characterized, our desired result follows. Our new proof uses our strong equidistribution theorem on affine subspaces of $\mathbb{K}^n$. An interesting point to note is that both the equidistribution theorem and the degree preserving lemma work only assuming that the field characteristic is constant and that the involved affine constraints are of bounded weight, without any assumption on the field size.

1.4.2 Reed-Muller codes

For a received word $g : \mathbb{K}^n \to \mathbb{K}$ our goal is to upper bound $|\{f \in \mathcal{P}_d : \text{dist}(f, g) \leqslant \eta\}|$, where $\eta = \delta_{\mathbb{K}}(d) - \varepsilon$ for some $\eta > 0$ and $\mathcal{P}_d$ is the class $\{Q : \mathbb{K}^n \to \mathbb{K} : \deg(Q) \leqslant d\}$. The proof technique is similar in structure as [14]. We apply the weak regularity lemma (Corollary 40) to the received word $g : \mathbb{K}^n \to \mathbb{K}$ and reduce the problem to a structured word $g' : \mathbb{K}^n \to \mathbb{K}$. More specifically, whenever $\mathrm{dist}(f, g) \leqslant \eta$, we have $\mathrm{dist}(f, g') \leqslant \eta + \varepsilon/2$. From here, we first express each function $f : \mathbb{K}^n \to \mathbb{K}$ as a linear combination of functions $f' : \mathbb{K}^n \to \mathbb{F}$. It can be then shown that the analysis in [14] works for functions $f' : \mathbb{K}^n \to \mathbb{F}$. A naive recombination of the $f' : \mathbb{K}^n \to \mathbb{F}$ to $f : \mathbb{K}^n \to \mathbb{K}$ gives us useful bounds only when $d < \mathrm{char}(|\mathbb{F}|)$. To circumvent this problem, we use our improved degree preserving theorem. This is crucial to our analysis as the technique of [14] can be used only to analyze the weight degree of polynomials which is not enough for the argument to work for arbitrary d and $|\mathbb{K}|$.

1.4.3 Polynomial decomposition

The algorithm and its analysis follows the lines of [6, 12]. Given a polynomial $P : \mathbb{K}^n \to \mathbb{K}$ (where $|\mathbb{K}|$ is bounded), we consider the collection of polynomials $\{\mathrm{Tr}(\alpha_1 P), \ldots, \mathrm{Tr}(\alpha_r P)\}$ where $\alpha_1, \ldots, \alpha_r \in \mathbb{K}$ are linearly independent. We regularize this collection into a pseudo-random polynomial factor and set one variable to 0 such that the degrees of the polynomials do not change. We then recursively solve the problem on $n-1$ variables and then apply a lifting procedure to get a decomposition for the original problem. A naive analysis of the lifting procedure over non-prime fields requires that $\deg(P) < \mathrm{char}(\mathbb{F})$. In order to get around this, we use our improved degree preserving theorem which applies for arbitrary degrees.

1.4.4 Testing affine-invariant properties

Suppose $\mathcal{P} \subseteq \{\mathbb{K}^n \to [R]\}$ is a locally characterized affine-invariant property (where R and $\mathrm{char}(\mathbb{K})$ are bounded but $n|\mathbb{K}|$ is growing). Our proof follows the lines of [11, 10, 9]. Suppose f is far from $\mathcal{P}$. We first identify a low-rank function close to f in an appropriate Gowers norm which also contains the violation that f contains. Here, low rank is with respect to a collection $\mathcal{B}$ of non-classical polynomials mapping to $\mathbb{T}$. We then investigate the distribution of $\mathcal{B}$ on the affine constraint that f violates. Since these are affine with respect to $\mathbb{K}^n$, we need to use our strong equidistribution theorem. The rest of the proof proceeds along the same lines as [9].

Because the proof of Theorem 6 is very analogous to that in [9] (except for the use of the new equidistribution theorem) and requires significant additional notation, we omit it here.

1.5 Some Open Questions

We conclude the introduction by giving a list of open questions suggested by this line of work:

- In our dichotomy result, can we show that if $P : \mathbb{K}^n \to \mathbb{K}$ is a polynomial of *weight degree* d, then either a character of it is unbiased or P is expressible in terms of other polynomials of weight degrees less than d? In the current form of the theorem, d is the degree of P.
- Can we show the dichotomy theorem for non-classical polynomials P? Together with the first item, we would then be able to iteratively regularize collections of polynomials over finite fields of growing size. Such a procedure would help resolve the list decoding radius for Reed-Muller codes over such fields, for instance.
- Can we improve the bound on the list size for Reed-Muller codes, compared to what we obtain in Theorem 2 or what is obtained in [13]?
- Can we use higher-order Fourier analysis to investigate the list-decoding radius for other codes, most notably, the Reed-Solomon code or the lifted Reed-Muller codes [32]?

- Is the notion of bounded weight characterization of affine-invariant properties an artifact of our proof or is it indeed linked to testability?

1.6 Organization

We formally define some notions like polynomials, bias, Gowers norm, and rank in Section 2. In Section 3, we show the bias versus rank dichotomy for non-classical polynomials. In Section 4, we show the equidistribution results for polynomials over subspaces. The next two sections, Sections 5 and 6, describe how the new tools can be used to prove the results for Reed-Muller list-decoding radius and algorithmic polynomial decomposition.

2 Preliminaries

Fix a prime number $p \geqslant 2$. Let $\mathbb{F}$ be the finite field of order p, and let $\mathbb{K}$ be a finite field of characteristic p. Let r denote the dimension of $\mathbb{K}$ as a vector space over $\mathbb{F}$; so, $|\mathbb{K}| = p^r$. Note that r is a parameter and may not be held constant.

Let $\mathrm{Tr} : \mathbb{K} \to \mathbb{F}$ denote the trace function: $\mathrm{Tr}(x) = x + x^p + x^{p^2} + \cdots + x^{p^{r-1}}$ for $x \in \mathbb{K}$. Recall that $\{x \to \mathrm{Tr}(ax) : a \in \mathbb{K}\}$ is in bijection with the set of all $\mathbb{F}$-linear maps from $\mathbb{K}$ to $\mathbb{F}$.

For every $\mathbb{K}$ of order r over $\mathbb{F}$, fix a choice of r linearly independent field elements $\alpha_1, \alpha_2, \dots, \alpha_r \in \mathbb{K}$. Then, there exists a dual basis, $\beta_1, \beta_2, \dots, \beta_r \in \mathbb{K}$ such that any $x \in \mathbb{K}$ can be written as

$$x = \sum_{i=1}^{r} \beta_i \mathrm{Tr}(\alpha_i x) \tag{1}$$

In particular, $\mathrm{Tr}(\alpha_i \beta_j)$ equals 1 if $i = j$ and 0 otherwise.

Let $\mathbb{T}$ denote the torus $\mathbb{R}/\mathbb{Z}$. This is an abelian group under addition. . For an integer $k \geqslant 0$, let $\mathbb{U}_k := \frac{1}{p^k}\mathbb{Z}/\mathbb{Z}$. Note that $\mathbb{U}_k$ is a subgroup of $\mathbb{T}$. Let $\iota : \mathbb{F} \to \mathbb{U}_1$ be the bijection $\iota(a) = \frac{|a|}{p} \pmod 1$. This map naturally extends to $\kappa : \mathbb{K} \to \mathbb{U}_1[Z_1, Z_2, \dots Z_r]$ where $Z_1, \dots, Z_r$ are formal variables, by the bijection $\kappa(x) = \sum_{i=1}^{r} \iota(\mathrm{Tr}(\alpha_i x)) \cdot Z_i$.

Let $e : \mathbb{T} \to \mathbb{C}$ be the function $e(x) = e^{2\pi i x}$. By abuse of notation, we sometimes write $e(x)$ for $x \in \mathbb{F}$ to mean $e(\iota(x))$.

2.1 Classical and Non-Classical Polynomials

We start by defining *classical polynomials.*

▶ **Definition 8.** (Classical polynomials over $\mathbb{K}$). We say that $P : \mathbb{K}^n \to \mathbb{K}$ is a *classical polynomial* of *degree* $\leqslant d$ if there exist coefficients $\{c_{i_1,\dots,i_n} \in \mathbb{K} : i_1, \dots, i_n \geqslant 0\}$ such that for all $x \in \mathbb{K}^n$:

$$P(x) = \sum_{i_1,\dots,i_n \geqslant 0, \sum_j i_j \leqslant d} c_{i_1,\dots,i_n} x_1^{i_1} \cdots x_n^{i_n}$$

where $c_{i_1,\dots,i_n}, x_1, \dots, x_n \in \mathbb{K}$.

As discussed above, there is a bijection $\kappa : \mathbb{K} \to \mathbb{T}[Z_1, \dots, Z_r]$ where $Z_1, \dots, Z_r$ are formal variables. This bijection carries a classical polynomial $P : \mathbb{K}^n \to \mathbb{K}$ of degree $\leqslant d$ to a function of the form:

$$\kappa(P(x)) = \sum_{j=1}^{r} \sum_{\substack{0 \leqslant i_1,\ldots,i_n: \\ \sum_j i_j \leqslant d}} \left(\frac{|\mathrm{Tr}(\alpha_j \cdot c_{i_1,\ldots,i_n} \cdot x_1^{i_1} \cdots x_n^{i_n})|}{p} \pmod 1 \right) \cdot Z_j$$

where $c_{i_1,\ldots,i_n} \in \mathbb{K}$. *Non-classical polynomials* are defined to map to $\mathbb{T}[Z_1, \ldots, Z_r]$ and have a similar representation.

▶ **Definition 9** (Non-classical polynomials over $\mathbb{K}$). We say that $Q : \mathbb{K}^n \to \mathbb{T}[Z_1, \ldots, Z_r]$ is a *non-classical polynomial* of *degree* $\leqslant d$ and *height* $\leqslant k$ if Q can be written as:

$$Q(x) = \sum_{j=1}^{r} \left(\gamma_j + \sum_{\ell=0}^{k} \sum_{\substack{0 \leqslant i_1,\ldots,i_n < p^r: \\ \sum_j i_j \leqslant d-\ell(p-1)}} \left(\frac{|\mathrm{Tr}(\alpha_j \cdot c_{i_1,\ldots,i_n,\ell} \cdot x_1^{i_1} \cdots x_n^{i_n})|}{p^{\ell+1}} \pmod 1 \right) \right) \cdot Z_j$$

for some $c_{i_1,\ldots,i_n,\ell} \in \mathbb{K}$ and $\gamma_j \in \mathbb{T}$.

Crucially, note that the coefficients $c_{i_1,\ldots,i_n,\ell}$ do not depend on j. Also, observe that non-classical polynomials of height 0 correspond to classical polynomials and that if $\mathbb{K} = \mathbb{F}$, this definition is identical to the one in [48].

In many parts of this paper, we will speak of non-classical polynomials $P : \mathbb{K}^n \to \mathbb{T}$. More precisely, this means that we identify $\mathbb{K}$ with $\mathbb{F}^{rn}$, and P is actually a non-classical polynomial over $\mathbb{F}$. In particular, it has the form:

$$P(x_1, \ldots, x_n) = \alpha + \sum_{\ell=0}^{k} \sum_{\substack{0 \leqslant d_{i,j} < p \ \forall i \in [n], j \in [r]: \\ \sum_{i=1}^{n} \sum_{j=1}^{r} d_{i,j} \leqslant d-k(p-1)}} \frac{c_{d_{1,1},\ldots,d_{n,r},k} \prod_{i=1}^{n} \prod_{j=1}^{r} |\mathrm{Tr}(\alpha_j x_i)|^{d_{i,j}}}{p^{\ell+1}} \pmod 1$$

where $\alpha \in \mathbb{T}$ and $c_{d_{1,1},\ldots,d_{n,r}} \in \{0, 1, \ldots, p-1\}$

A particular type of classical polynomial plays an important role in our analysis.

▶ **Definition 10** (Classical, symmetric, multilinear (CSM) forms). We say that a polynomial $T : (\mathbb{K}^n)^d \to \mathbb{K}$ is in $\mathrm{CSM}_d(\mathbb{K}^n)$ if $T(h_1, \ldots, h_d)$ is of the form

$$T(h_1, \ldots, h_d) = \sum_{i_1,\ldots,i_d \in [n]} c_{\{i_1,\ldots,i_d\}} h_{1,i_1} \cdots h_{d,i_d}$$

where $c_{\{i_1,\ldots,i_d\}} \in \mathbb{K}$ and $h_1, \ldots, h_d \in \mathbb{K}^n$. T satisfies the following properties

1. Multilinear: Each term in T has the form above.
2. Symmetric: T is invariant with regards to permutations of $h_1, \ldots, h_d$.
3. Classical: $T(h_1, \ldots, h_d)$ vanishes whenever at least p of the $h_1, \ldots, h_d \in \mathbb{K}^n$ are equal.

2.2 Additive Derivatives and Weight Degree

Although polynomials are defined in the above section in terms of their global representation, we will often be interested in local constraints obeyed by functions.

▶ **Definition 11** (Additive derivative and Weight degree). Given a function $f : \mathbb{K}^n \to \mathbb{T}[Z_1, \ldots, Z_r]$, its *additive derivative in direction* $h \in \mathbb{K}^n$ is $D_h f : \mathbb{K}^n \to \mathbb{T}[Z_1, \ldots, Z_r]$, given by

$$D_h f(x) = f(x+h) - f(x).$$

A function $P : \mathbb{K}^n \to \mathbb{T}[Z_1, \dots, Z_r]$ is said to have *weight degree* $\leqslant w$ if for all $x, h_1, h_2, \dots, h_{w+1} \in \mathbb{K}^n$,

$$D_{h_1} D_{h_2} \cdots D_{h_{w+1}} P(x) = 0. \tag{2}$$

If $f : \mathbb{K}^n \to \mathbb{T}[Z_1, \dots, Z_r]$ and $f(x) = \sum_{i=1}^r f_i(x) \cdot Z_i$, then it is clear that f has weight degree $\leqslant w$ if and only if each $f_i : \mathbb{K}^n \to \mathbb{T}$ is a non-classical polynomial of degree $\leqslant w$ (in the sense of [48]). This is because for functions mapping from a vector space over $\mathbb{F}$ to $\mathbb{T}$, the notion of degree and weight degree coincide. In particular:

▶ **Fact 12.** *The degree and weight degree of any non-classical polynomial $P : \mathbb{K}^n \to \mathbb{T}$ are equal.*

But what about the relation between degree and weight degree for functions mapping to $\mathbb{T}[Z_1, \dots, Z_r]$? Here, we can make two remarks.

▶ Remark. As mentioned above, a bound of w for the weight degree of a function $f = \sum_i f_i \cdot Z_i$ means that each f_i is individually of (weight) degree $\leqslant w$, but it does not impose any relationship whatsoever between the different f_i's. On the other hand, in Definition 9, the different f_i's are all determined by the same set of coefficients in $\mathbb{K}$.

▶ Remark. If $P : \mathbb{K}^n \to \mathbb{T}[Z_1, \dots, Z_r]$ is a non-classical polynomial, then its weight degree is the maximum weight degree of any of its terms, where the weight degree of a term $\frac{\mathrm{Tr}(\alpha_j c_{i_1,\dots,i_n,\ell} x_1^{i_1} \cdots x_n^{i_n})}{p^{\ell+1}}$ is $\ell(p-1) + \mathsf{wt}(i_1) + \mathsf{wt}(i_2) + \cdots + \mathsf{wt}(i_n)$ and $\mathsf{wt}(i)$ for $0 \leqslant i < p^r$ is the sum of the r digits of i in its p-ary expansion. Hence, if every individual degree i_k is less than p, then the weight degree equals the degree of the polynomial. Also, clearly, the weight degree is always at most the degree for any non-classical polynomial.

2.3 Bias and Gowers norm

▶ **Definition 13.** The *bias* of a function $f : \mathbb{K}^n \to \mathbb{K}$ is defined as $\mathrm{bias}(P) = |\mathbb{E}_{x \in \mathbb{K}^n} e(\mathrm{Tr}(P(x)))|$.

Here, we could have used any non-trivial additive character instead of the trace, but we choose this definition for concreteness (without any loss of generality). The *Gowers norm* of a function measures the bias of its iterated derivative.

▶ **Definition 14** (Gowers norm). Given a function $P : \mathbb{K}^n \to \mathbb{K}$ and an integer $d \geqslant 1$, the *Gowers norm of order d* for P is given by

$$\|P\|_{U^d} = \left| \mathop{\mathbb{E}}_{h_1,\dots,h_d,x \in \mathbb{K}^n} [e(D_{h_1} \cdots D_{h_d} \mathrm{Tr}(P(x)))] \right|^{1/2^d}.$$

Note that as $\|f\|_{U^1} = \mathrm{bias}(f)$ the Gowers norm of order 1 is only a semi-norm. However for $d > 1$, it is not difficult to show that $\|\cdot\|_{U^d}$ is indeed a norm. Also, note that $\|P\|_{U^d} \geqslant \mathrm{bias}(P)$ for any $d \geqslant 1$.

We can also write the Gowers norm in a slightly different way which will be convenient for us.

▶ **Definition 15** (Derivative polynomial). If $P : \mathbb{K}^n \to \mathbb{K}$ is a classical polynomial of degree d (i.e., some term of P has total degree exactly d), then let the *derivative polynomial* be $DP : \mathbb{K}^{nd} \to \mathbb{K}$ defined as $DP(h_1, \dots, h_d) = D_{h_1} D_{h_2} \cdots D_{h_d} P(x)$, which is independent of x.

▶ **Lemma 16.** *Let $P : \mathbb{K}^n \to \mathbb{K}$ be a classical polynomial of degree d. Then, $DP \in \mathrm{CSM}_d(\mathbb{K}^n)$, and $\|P\|_{U^d}^{2^d} = \mathrm{bias}(DP)$.*

Note that if d were the weight degree instead of the degree in Lemma 16, then DP would be multilinear in the sense that for any $i \in [d]$, $DP(h_i + h_i', (h_j)_{j \neq i}) = DP(h_i, (h_j)_{j \neq i}) + DP(h_i', (h_j)_{j \neq i})$, but individual variables could have degree more than 1 (any power of p) in DP and so could not be in $\mathrm{CSM}_d(\mathbb{K}^n)$ according to Definition 10.

2.4 Rank

▶ **Definition 17.** Let $P : \mathbb{K}^n \to \mathbb{K}$ be a classical polynomial of weight degree w. The $\mathbb{K}$-*rank* of P is the smallest integer c such that there exist functions $Q_1, \ldots, Q_c : \mathbb{K}^n \to \mathbb{T}[Z_1, \ldots, Z_r]$ of weight degree $< w$ and a function $\Gamma : \mathbb{T}[Z_1, \ldots, Z_r]^c \to \mathbb{K}$ such that $P(x) = \Gamma(Q_1(x), \ldots, Q_c(x))$.

We will often restrict to the case when p and r are constant. In this case, we can look at the collection of polynomials $\mathcal{P} = \{\mathrm{Tr}(\alpha_1 P), \ldots, \mathrm{Tr}(\alpha_r P)\}$. These are non-classical polynomials of degree w from $\mathbb{F}^{rn}$ to $\mathbb{T}$. Then, upto a factor of r, the minimum $\mathbb{F}$-rank of any nonzero linear combination of the polynomials in $\mathcal{P}$ is at most the $\mathbb{K}$-rank of P.

3 Inverse Theorem for Classical Polynomials

In this section, we establish the Gowers Inverse theorem for polynomial phases over growing field sizes.

▶ **Theorem 18.** *Let $d, p, r, s \in \mathbb{N}$, and $\mathbb{K}$ be a field extension of $\mathbb{F} = \mathbb{F}_p$ with $[\mathbb{K} : \mathbb{F}] = r$. Then, there exists $c = c^{18}(d, s)$ such that the following is true. Consider any classical polynomial $P : \mathbb{K}^n \to \mathbb{K}$ of degree $\leqslant d$ such that $\|P\|_{U^d} \geqslant |\mathbb{K}|^{-s}$. Then, there exist non-classical polynomials $R_1, \ldots, R_c : \mathbb{K}^n \to \mathbb{T}[Z_1, \ldots, Z_r]$ of weight degrees $\leqslant d-1$ and a function $\Gamma : \mathbb{T}[Z_1, \ldots, Z_r]^c \to \mathbb{K}$ such that $P = \Gamma(R_1, \ldots, R_c)$.*

For context, recall Deligne's multivariate generalization of Weil's bound which implies that if $d = \deg(P)$ is a constant, and $\mathrm{bias}(P) > |\mathbb{K}|^{-1/2}$, then P must have weight degree 0. Our theorem shows the structure of constant-degree polynomials when their bias is smaller but still lower bounded by an inverse polynomial in $\mathbb{K}$.

An immediate corollary of Theorem 18 is that:

▶ **Corollary 19.** *Let $d, p, r, s \in \mathbb{N}$, and $\mathbb{K}$ be a field extension of $\mathbb{F} = \mathbb{F}_p$ with $[\mathbb{K} : \mathbb{F}] = r$. Then, there exists $c = c^{18}(d, s)$ such that the following is true. Consider any classical polynomial $P : \mathbb{K}^n \to \mathbb{K}$ of degree and weight degree d such that $\mathrm{bias}(P) \geqslant |\mathbb{K}|^{-s}$. Then, $\mathrm{rank}(P) \leqslant c^{18}(d, s)$.*

It is open how to remove the restriction that the degree and weight degree of P are equal. We now go on to the proof of Theorem 18.

Proof. Let DP denote the derivative polynomial of P. By Lemma 16,

$$\mathrm{bias}(\mathrm{Tr}(DP)) \geqslant |\mathbb{K}|^{-s2^d}$$

If the weight degree of P is strictly less than d, we are already done. If not, then DP is a nonzero polynomial. We will show that there exist non-classical polynomials $Q_1, \ldots, Q_c : \mathbb{K}^n \to \mathbb{T}[Z_1, \ldots, Z_r]$ of weight degree $< d$ such that for every $i \in [r]$,

$$\mathrm{Tr}(\alpha_i DP) = D\Gamma_i(Q_1, \ldots, Q_c) \tag{3}$$

for some function Γ_i mapping to $\mathbb{F}$. Therefore, because trace and derivative commute and using (1):

$$DP = D\left(\sum_{i=1}^{r} \Gamma_i(Q_1, \dots, Q_c) \cdot \beta_i\right)$$

In other words, P and $\sum_{i=1}^{r} \Gamma_i(Q_1, \dots, Q_c) \cdot \beta_i$ differ by a polynomial of weight degree $\leqslant d-1$, proving the theorem.

Our proof for (3) will heavily use the structure of CSM forms. To this end, let us make a couple of definitions and observations about operations on CSM forms.

▶ **Definition 20** (Concatenation). Let $P \in \mathrm{CSM}_k(\mathbb{K}^n)$ and $Q \in \mathrm{CSM}_\ell(\mathbb{K}^n)$ for integers $k, \ell \geqslant 1$. Then the *concatenation operator* $P * Q \in \mathrm{CSM}_{k+\ell}(\mathbb{K}^n)$ is defined as

$$P * Q(y_1, \dots, y_{k+\ell}) = \sum_{A \subseteq [k+\ell], |A|=k} P((y_i)_{i \in A}) Q((y_i)_{i \in [k+\ell] \setminus A})$$

▶ **Lemma 21.** *Given two classical polynomials $P : \mathbb{K}^n \to \mathbb{K}$ and $Q : \mathbb{K}^n \to \mathbb{K}$, $D(P \cdot Q) = DP * DQ$.*

▶ **Definition 22** (Symmetric Power). Let $d \geqslant 2$ and $P \in \mathrm{CSM}_d(\mathbb{K}^n)$, then for $m \geqslant 1$, the *symmetric power* $\mathrm{Sym}^m(P) \in \mathrm{CSM}_{md}(\mathbb{K}^n)$ is defined as

$$\mathrm{Sym}^m(P)(h_1, \dots, h_{md}) = \sum_{\mathcal{A}} \prod_{A \in \mathcal{A}} P((h_i)_{i \in A})$$

where the sum is over all possible partitions $\mathcal{A}$ of $\{1, \dots, md\}$ into m-equal sized subsets.

▶ Remark. Note that $d \geqslant 2$ in the definition of symmetric power. If $d = 1$, then the symmetric power need not satisfy the third condition in Definition 10 and hence may not be CSM.

▶ Remark. Below, we'll apply the symmetric power operation to the trace of a $\mathrm{CSM}_d(\mathbb{K}^n)$ form, rather than to the form directly. However, note that the trace of a CSM is also classical, symmetric and multilinear, though now mapping to $\mathbb{F}$.

Now, we continue with the main thread of our proof. Our first step shows the structure of high-bias CSM forms.

▶ **Theorem 23** (Analog of Theorem 6.6 in [48]). *Suppose $d \geqslant 2$ and $s \geqslant 1$. Let $T \in \mathrm{CSM}_d(\mathbb{K}^n)$ such that* $\mathrm{bias}(T(h_1, \dots, h_d)) > |\mathbb{K}|^{-s}$. *Then, there exists a subspace $V \subseteq \mathbb{K}^n$ of codimension $\leqslant r^{23}(d, s)$ such that restricted to V^d,* $\mathrm{Tr}(T)$ *is a linear combination over $\mathbb{F}$ of at most $t^{23}(d, s)$ expressions of the form:*

$$\mathrm{Sym}^{m_1}(\mathrm{Tr}(S_1)) * \cdots * \mathrm{Sym}^{m_k}(\mathrm{Tr}(S_k))$$

for some $m_1, \dots, m_k \geqslant 1$ and $2 \leqslant d_1, \dots, d_k < d$ where $S_j \in \mathrm{CSM}_{d_j}(V')$ for $j \in [k]$ with $m_1 d_1 + \cdots + m_k d_k = d$.

Proof. Our proof is very close to the one by Tao and Ziegler [48]. Where they use the lemma by Bogdanov and Viola [16] to approximate a biased polynomial by its derivatives, we use the newer version of this result by Bhowmick and Lovett [13] that gives bounds independent of field size. By Lemma 3.1 of [13], for any $t \geqslant 1$, we obtain $Q_1, \dots, Q_c : \mathbb{K}^{nd} \to \mathbb{K}$ and $\Gamma : \mathbb{F}^c \to \mathbb{F}$ where $c = c^{BL}(d, s, t)$ such that:

$$\mathbf{Pr}_{x \in \mathbb{K}^n}[\mathrm{Tr}(T(x)) \neq \Gamma(\mathrm{Tr}(Q_1(x), \dots, \mathrm{Tr}(Q_c(x))))] \leqslant |\mathbb{K}|^{-t}.$$

Each Q_i here is an additive derivative of T. However, we want an exact representation of $\mathrm{Tr}(T)$ in terms of lower degree polynomials. Kaufmann and Lovett [37] showed that if t is large enough in terms of d, and if $\mathrm{Tr}(Q_1), \dots, \mathrm{Tr}(Q_c)$ form a *strongly regular* factor in the sense of [37], then in fact, $\mathrm{Tr}(T)$ is exactly a function of $\mathrm{Tr}(Q_1), \dots, \mathrm{Tr}(Q_c)$. We use the regularization procedure in [12] (Lemma 5.2), which iteratively replaces one of the polynomials in the current collection $\mathrm{Tr}(Q_1), \dots, \mathrm{Tr}(Q_c)$ with an additive derivative of one of the polynomials in a chosen direction. We do not repeat the definitions and proofs of these results as they closely follow previous work. We also note that we can always write any derivative of a trace of a CSM form in terms of traces of other CSM forms. This follows from the below claim as trace is linear:

▶ **Claim 24.** *Let $s > 0$ be an integer, $L \in \mathrm{CSM}_s(\mathbb{K}^n)$, and $a \in (\mathbb{K}^n)^s$. Then the additive derivative of L in direction a, D_aL, can be written as a linear combination of $2^s - 1$ polynomials $(Q_S)_{S \subset [s], S \neq \emptyset}$, where $Q_S \in \mathrm{CSM}_{s-|S|}(\mathbb{K}^n)$ and $c_S \geqslant 1$.*

Proof. We can write

$$D_aL(h_1, \dots, h_s) = L(h_1 + a_1, \dots, h_s + a_s) - L(h_1, \dots, h_s) = \sum_{\substack{S \subset [s] \\ S \neq \emptyset}} L((h_i)_{i \notin S}, (a_i)_{i \in S})$$

where the second equality follows from multilinearity of T. Now letting $Q_S := L((h_i)_{i \notin S}, (1^n)^{s-|S|})$ we have that $Q_S \in \mathrm{CSM}_{s-|S|}(\mathbb{K}^n)$. ◀

The decomposition into concatenation of symmetric powers follows exactly as in [48]. ◀

Therefore, applying Theorem 23 with $T = DP$, we get that for a bounded index subspace V, $\mathrm{Tr}(DP)$ restricted to V^d is a linear combination of a bounded number of expressions of the form:

$$\mathrm{Sym}^{m_1}(\mathrm{Tr}(S_1)) * \cdots * \mathrm{Sym}^{m_k}(\mathrm{Tr}(S_k)) \,.$$

We next note that since $DP \in \mathrm{CSM}_d(\mathbb{K}^n)$, $\mathrm{Tr}(\alpha_i DP)(h_1, h_2, \dots, h_d) = \mathrm{Tr}(DP)(\alpha_i h_1, h_2, \dots, h_d)$. Also, because $S_1, \dots, S_k$ are each CSM, $S_j(\alpha_i h_1, h_2, \dots, h_d)$ equals $\alpha_i S_j(h_1, h_2, \dots, h_d)$ if S_j depends on h_1 and equals $S_j(h_1, h_2, \dots, h_d)$ otherwise. Hence, for every $i \in [r]$, we get that restricted to the subspace V, $\mathrm{Tr}(\alpha_i DP)$ is a linear combination of a bounded number of "monomials" of the form:

$$\mathrm{Sym}^{m_1}(\mathrm{Tr}(\gamma_{i,1} S_1)) * \cdots * \mathrm{Sym}^{m_k}(\mathrm{Tr}(\gamma_{i,k} S_k)) \tag{4}$$

where $\gamma_{i,1}, \dots, \gamma_{i,k} \in \mathbb{K}$. Crucially, $S_1, \dots, S_k$ in all of the above "monomials" are independent of α_i.

Consider any one "monomial" of the form in (4), and we show that there exist non-classical polynomials $Q_1, \dots, Q_c : \mathbb{K}^n \to \mathbb{T}[Z_1, \dots, Z_r]$ of weight degrees $< d$ such that

$$\mathrm{Sym}^{m_1}(\mathrm{Tr}(\gamma_{i,1} S_1)) * \cdots * \mathrm{Sym}^{m_k}(\mathrm{Tr}(\gamma_{i,k} S_k)) = D\Delta_i(Q_1, \dots, Q_k) \tag{5}$$

for some function $\Delta_i : \mathbb{T}[Z_1, \dots, Z_r]^c \to \mathbb{F}$. Restricted to V^d, our desired form (3) then follows from linearity.

We first show a converse to Lemma 16 which resolves the situation when $m_j = 1$.

▶ **Lemma 25.** *For positive integer d, suppose $S \in \mathrm{CSM}_d(\mathbb{K}^n)$. Then there is a degree-d classical polynomial $Q : \mathbb{K}^n \to \mathbb{K}$ such that $DQ = S$.*

Proof. Tao and Ziegler (Lemma 4.5, [48]) show the same result for CSM forms over $\mathbb{F}^n$, and their proof works without any change. ◀

The next lemma shows that we can integrate symmetric powers of traces of CSM's in terms of non-classical polynomials.

▶ **Lemma 26.** *Let $d \geqslant 2$ and $m \geqslant 1$, $\gamma_1, \dots, \gamma_r \in \mathbb{K}$, and let $S \in \mathrm{CSM}_d(\mathbb{K}^n)$. Then, there exists a classical polynomial $W : \mathbb{K}^n \to \mathbb{K}$ of weight degree $\leqslant md$ such that $D\mathrm{Tr}(\alpha_i W) = \mathrm{Sym}^m(\mathrm{Tr}(\gamma_i S))$ for all $i \in [r]$. Moreover, if $m \geqslant 2$, then W is a function of a non-classical polynomial of degree $< md$.*

We defer the proof of Lemma 26 to Section 3.1 but we first explain how to complete the proof of Theorem 18. Applying Lemma 26 on each term in the concatenation product in (5), we get for all $j \in [k]$, classical polynomials $W_j : \mathbb{K}^n \to \mathbb{K}$ of weight degree $\leqslant m_j d_j$ such that $D\mathrm{Tr}(\alpha_i W_j) = \mathrm{Sym}^{m_j}(\mathrm{Tr}(\gamma_{i,j} S_j))$, so that the expression in (4) is the derivative polynomial of $U = \prod_{j=1}^k \mathrm{Tr}(\alpha_i W_j)$ by Lemma 21. Note that if $k > 1$, then U is already a function of more than one classical polynomial of weight degree $< d$. Otherwise, if $k = 1$, then $m_1 \geqslant 2$ (as $d \geqslant 2$ and $d_1 < d$), and so, U is a function of the non-classical polynomial of degree (and hence, weight degree) $< d$ determining W_1 that is guaranteed to exist by Lemma 26.

We have proved Theorem 18 when all the variables are drawn from V, a subspace of $\mathbb{K}^n$ of co-dimension $t \leqslant t^{23}$. We have shown that we have a degree-d classical polynomial S measurable in non-classical polynomials $\{\tilde{R}_1, \dots, \tilde{R}_C\}$ of weight degrees $\leqslant d-1$ such that $DP = DS$ on the bounded index subspace V. We can extend the last statement to the subspace $\mathbb{K}^n$ by using a simple derivative trick. Suppose $h'_1, \cdots, h'_t$ are representatives of the quotient group $\mathbb{K}^n/V$. Thus for any $x \in \mathbb{K}^n$, we can have a $i \in [K]$ such that $x - h'_i \in V$. We can write

$$S(x) = S(x - h'_i) - D_{-h'_i} S(x)$$

for x and note that $\deg(D_{-h'_i} S) < d$. This implies that over $\mathbb{K}^n$, S is measurable in $\{D_{-h'_1} S, \dots, D_{-h'_t} S, \tilde{R}_1, \dots, \tilde{R}_C : \mathbb{K}^n \to \mathbb{T}[Z_1, \dots, Z_c]\}$ and $DP = DS$. Now, by letting $Q = S$ and $P' = P - S$, then $DP' = 0$, meaning that weighted degree of P' is less than d, which concludes the proof. ◀

▶ Remark. It is worth noticing that in fact, the proof shows that for every $i \in [r]$, $\mathrm{Tr}(\alpha_i P)$ also has $\mathbb{F}$-rank bounded by $c^{18}(d, s)$, provided that P has degree and weight degree d.

3.1 Proof of Lemma 26

Proof. Our proof follows the outline of the proof of Lemma 6.4 in [48] but with some modifications.

Apply Lemma 25 to get that there is a classical polynomial $R : \mathbb{K}^n \to \mathbb{K}$ of degree d such that $DR = S$. Let $M \geqslant 0$ be an integer such that $p^M \leqslant m < p^{M+1}$. There exists a degree-$(d + M(p-1))$ polynomial $\tilde{R}_M : \mathbb{K}^n \to \mathbb{T}[Z_1, \dots, Z_r]$ such that $p^M \cdot \tilde{R}_M = \kappa(R)$. And then we pull down the polynomial $\tilde{R}_M$ to the cyclic group $(\mathbb{Z}/p^{M+1}\mathbb{Z})^r$ and we obtain a degree-$(d + M(p-1))$ polynomial $R_M : \mathbb{K}^n \to \left(\mathbb{Z}/p^{M+1}\mathbb{Z}\right)^r$ such that $R_M = R \pmod p$.

Define $\delta_{i,j} = |\mathrm{Tr}(\gamma_i \alpha_j)|$ where $i, j \in [r]$, and thus $\gamma_i = \sum_j \delta_{i,j} \beta_j$. Let

$$W = \sum_{i=1}^r \left(\binom{\sum_{j=1}^r \delta_{i,j} (R_M)_j}{m} \pmod p \right) \cdot \beta_i$$

where $(R_M)_j$ denotes the jth component of R_M. Define

$$W_{\alpha_i} = \mathrm{Tr}(\alpha_i W) = \binom{\sum_{j=1}^{r} \delta_{i,j}(R_M)_j}{m} \pmod p$$

We claim that

1. $\deg(W_{\alpha_i}) \leqslant md$.
2. $DW_{\alpha_i} = \mathrm{Sym}^m(\gamma_{i,j}DR)$.

Parts 1 and 2 together imply that the weight degree of (W) is at most md. Fix any $i \in [r]$ and define $S = \sum_{j=1}^{r} \delta_{i,j}(R_M)_j$. The last part of the theorem follows from the fact that $d + M(p-1) < md$ when $m > 1$ and that Q can be expressed as a function of $\tilde{R}_M$ which is of degree $d + M(p-1)$. Part 1 will be a special case of the following claim, that is, when $j = 0$ and $m' = m$.

▶ **Lemma 27.** *Let $j \geqslant 0$ and $m' \leqslant m$ be some parameters. Then*

$$\deg\left(\binom{D_{h_1}\cdots D_{h_j}S}{m'} \pmod p\right) \leqslant d - j + (m'-1)\cdot\max(d-j,1).$$

Proof. We break the claim into two cases:

Case i. $(d - j + (m'-1)\cdot\max(d-j,1) < 0)$: In particular, this implies that $m' < j - d$. We need to show that $\binom{D_{h_1}\cdots D_{h_j}S}{m'}$ is divisible by p. Since $\deg(S) = d + M(p-1)$ then $\deg(D_{h_1}\cdots D_{h_j}S) \leqslant d - j + M(p-1)$ for any $h_1, \ldots, h_j \in \mathbb{K}^n$. And, it is divisible by p^{a+1} whenever $0 \leqslant a \leqslant M$ and $d - j + a(p-1) < 0$. In particular, if we choose $a = \lfloor \frac{m'-1}{p-1} \rfloor$, then $D_{h_1}\cdots D_{h_j}S$ is divisible by $p^{\lfloor \frac{m'-1}{p-1} \rfloor + 1}$. Observe that $\binom{n}{m} \pmod p$ is divisible by p if n is divisible by p^a and $m < p^a$. Since $m' < p^{\lfloor \frac{m'-1}{p-1} \rfloor + 1} = p^{a+1}$, we obtain our claim.

Case ii. $(d - j + (m'-1)\cdot\max(d-j,1) \geqslant 0)$: We will prove this by downward induction on j. The claim is already true for sufficiently large values of j, so we assume inductively that the claim is proven for all larger values of j; and for fixed j, we assume inductively that the claim is proven for all smaller m'. It suffices to show that the expression

$$deg\left(D_{h_{j+1}}\binom{D_{h_1}\cdots D_{h_j}S}{m'} \pmod p\right) \leqslant (d - j + (m'-1)\cdot\max(d-j,1) - 1$$

holds $\forall h_{j+1} \in \mathbb{K}^n$. We will use the combinatorial identity $\binom{r+s}{m} = \sum_{i=0}^{m} \binom{r}{i}\binom{s}{m-i}$ and then we see that

$$D_h\binom{F}{m} = \sum_{i=1}^{m} \binom{D_hF}{i}\binom{F}{m-i}$$

for $h \in \mathbb{K}^n$ and $F : \mathbb{K}^n \to \mathbb{Z}/p^{M+1}\mathbb{Z}$. We can therefore write

$$D_{h_{j+1}}\binom{D_{h_1}\cdots D_{h_j}S}{m'} = \sum_{i=1}^{m'} \binom{D_{h_1}\cdots D_{h_{j+1}}S}{i}\binom{D_{h_1}\cdots D_{h_j}S}{m'-i}$$

where the both sides of the above equality is over mod p. Now for each summand, we apply the two induction hypothesis and conclude that the degree of the first factor in the right-hand side is $\leqslant d - (j+1) + (i-1)\max(d-(j+1),1)$ and the degree of the second factor in the right-hand side is $\leqslant d-j+(m'-i-1)\max(d-j,1)$ and thus the degree of the right-hand side is atmost

$$(d-(j+1)+(i-1)\max(d-(j+1),1))+(d-j+(m'-i-1)\max(d-j,1)) \\ \leqslant d-j+(m'-1)\max(d-j,1)-1$$

whenever $i \geqslant 1$ (by handling the cases $d-j>1$ and $d-j \leqslant 1$ separately), and this concludes the claim that $\deg(W_{\alpha_i}) \leqslant md$. ◀

Now we prove the part 2. We know that

$$D_h \binom{S}{m} = \sum_{i=1}^{m} \binom{D_h S}{i} \binom{S}{m-i}$$

for any $h \in \mathbb{K}^n$. By the above computations, the polynomial $\binom{D_h S}{i}\binom{S}{m-i}$ has degree

$$\leqslant d-1+(i-1)(d-1)+d+(m-i-1)d = md-i.$$

In particular, all the terms of $i>1$ have degree $< mk-1$ and thus will not contribute to $D\binom{S}{m}$. The $i=1$ term can be simplified as $D_h \mathrm{Tr}(\gamma_i R)\binom{S}{m-1}$ by using the equality

$$\mathrm{Tr}(\gamma_i \cdot R) = \mathrm{Tr}\big((\sum_{i=1}^{r} \delta_{i,j}\beta_i)\cdot(\sum_{j=1}^{r}(R)_j\alpha_j)\big) = \sum_{j=1}^{r}\delta_{i,j}\cdot(R)_j$$

We conclude that

$$\begin{aligned} DW_{\alpha_i}(h_1,\ldots,h_{md}) &= D\big(D_{h_{md}}\binom{S}{m}\big)(h_1,\ldots,h_{md-1}) \\ &= D\big(\big(D_{h_{md}}\mathrm{Tr}(\gamma_i R)\big)\cdot\binom{S}{m-1}\big)(h_1,\ldots,h_{md-1}) \\ &= \big(D\big(D_{h_{md}}\mathrm{Tr}(\gamma_i R)\big) * D\binom{S}{m-1}\big)(h_1,\ldots,h_{md-1}) \\ &= \sum_{1\leqslant i_1<\cdots<i_{d-1}<md} D\big(D_{h_{md}}\mathrm{Tr}(\gamma_i R)\big)(h_{i_1},\ldots,h_{i_{d-1}},h_{md})\cdot D\binom{S}{m-1}(h_{j_1},\ldots,h_{j_{md-d}}) \end{aligned}$$

where $1 \leqslant j_1 < \cdots < j_{md-d} < md$ are such that $\{j_1,\ldots,j_{md-d}\} = \{1,\ldots,md-1\} \setminus \{i_1,\ldots,i_{d-1}\}$ and by Lemma 21, we have the second equality. Now, by induction on m, we have our claim 2. ◀

4 Equidistribution of regular factors

Our results in this section imply that a high rank collection of polynomials is "as random as possible", subject to the degree and depth bounds of its defining polynomials. We first make some necessary definitions.

4.1 Definitions

A *linear form on k variables* is a vector $L=(w_1,w_2,\ldots,w_k)\in\mathbb{K}^k$ that is interpreted as a function from $(\mathbb{K}^n)^k$ to $\mathbb{K}^n$ via the map $(x_1,\ldots,x_k)\mapsto w_1x_1+w_2x_2+\cdots+w_kx_k$. A linear form $L=(w_1,w_2,\ldots,w_k)$ is said to be *affine* if $w_1=1$. From now, linear forms will always be assumed to be affine. We define wt of a linear form $L=(w_1,\ldots,w_k)$ to be $\sum_{i=2}^{k}\mathsf{wt}(w_i)$.

We specify a partial order $\preceq$ among affine forms. We say $(w_1,\ldots,w_k)\preceq(w'_1,\ldots,w'_k)$ if $|\mathrm{Tr}(\alpha_j w_i)| \leqslant |\mathrm{Tr}(\alpha_j w'_i)|$ for all $i\in[k], j\in[r]$.

▶ **Definition 28** (Affine constraints)**.** An *affine constraint of size m on k variables* is a tuple $A = (L_1, \dots, L_m)$ of m affine forms $L_1, \dots, L_m$ over $\mathbb{F}$ on k variables, where: $L_1(x_1, \dots, x_k) = x_1$. Moreover, it is said to be *weight-closed* if for any affine form L belonging to A, if $L' \preceq L$, then L' also belongs to A.

Observe that a weight-closed affine constraint is of bounded size if and only if all its affine forms are of bounded weight.

Next, we define polynomial factors which play a big role in higher-order Fourier analysis. Here, we restrict ourselves to non-classical polynomials mapping to $\mathbb{T}$ (instead of $\mathbb{T}[Z_1, \dots, Z_r]$), essentially because throughout we mainly care about the case of constant r.

▶ **Definition 29** (Factor)**.** A *polynomial factor* $\mathcal{B}$ is a sequence of non-classical polynomials $P_1, \dots, P_C : \mathbb{K}^n \to \mathbb{T}$. We also identify it with the function $\mathcal{B} : \mathbb{K}^n \to \mathbb{T}^C$ mapping x to $(P_1(x), \dots, P_C(x))$. An *atom* of $\mathcal{B}$ is a preimage $\mathcal{B}^{-1}(y)$ for some $y \in \mathbb{T}^C$. When there is no ambiguity, we will in fact abuse notation and identify an atom of $\mathcal{B}$ with the common value $\mathcal{B}(x)$ of all x in the atom.

The *partition induced by* $\mathcal{B}$ is the partition of $\mathbb{K}^n$ given by $\left\{\mathcal{B}^{-1}(y) : y \in \mathbb{T}^C\right\}$. The *complexity* of $\mathcal{B}$, denoted $|\mathcal{B}|$, is the number of defining polynomials C. The *order* of $\mathcal{B}$, denoted $\|\mathcal{B}\|$, is the total number of atoms in $\mathcal{B}$. The *degree* of $\mathcal{B}$ is the maximum degree among its defining polynomials $P_1, \dots, P_C$.

Note that due to Definition 9, if $\mathcal{B}$ is defined by polynomials $P_1, \dots, P_C$,

$$\|\mathcal{B}\| = \prod_{i=1}^{C} p^{\mathrm{depth}(P_i)+1} .$$

From henceforth, since we will work only with non-classical polynomials $P : \mathbb{K}^n \to \mathbb{T}$, rank will denote their $\mathbb{F}$-rank.

▶ **Definition 30** (Rank and Regularity of Polynomial Factor)**.** Let $\mathcal{B}$ be a polynomial factor defined by the sequence $P_1, \dots, P_c : \mathbb{K}^n \to \mathbb{T}$ with respective depths $k_1, \dots, k_c$. Then, the rank of $\mathcal{B}$ is $\min_{(a_1,\dots,a_c)} \mathrm{rank}(\sum_{i=1}^{c} a_i P_i)$ where the minimum is over $(a_1, \dots, a_c) \in \mathbb{Z}^c$ such that $(a_1 \mod p^{k_1+1}, \dots, a_c \mod p^{k_c+1}) \neq (0, \dots, 0)$.

Given a polynomial factor $\mathcal{B}$ and a non decreasing function $r : \mathbb{Z}^+ \to \mathbb{Z}^+$, $\mathcal{B}$ is r-regular if $\mathcal{B}$ is of rank at least $r(|\mathcal{B}|)$.

▶ **Definition 31** (Semantic and Syntactic refinement)**.** Let $\mathcal{B}$ and $\mathcal{B}'$ be polynomial factors. A factor $\mathcal{B}'$ is a *syntactic refinement* of $\mathcal{B}$, denoted by $\mathcal{B}' \succeq_{syn} \mathcal{B}$ if the set of polynomials defining $\mathcal{B}$ is a subset of the set of polynomials defining $\mathcal{B}'$. It is a *semantic refinement*, denoted by $\mathcal{B}' \succeq_{sem} \mathcal{B}$ if for every $x, y \in \mathbb{K}^n$, $\mathcal{B}'(x) = \mathcal{B}'(y)$ implies $\mathcal{B}(x) = \mathcal{B}(y)$. Clearly, a syntactic refinement is also a semantic refinement.

Our next lemma is the workhorse that allows us to convert any factor into a regular one.

▶ **Lemma 32** (Polynomial Regularity Lemma)**.** *Let $r : \mathbb{Z}^+ \to \mathbb{Z}^+$ be a non-decreasing function and $d > 0$ be an integer. Then, there is a function $C_{32}^{(r,d)} : \mathbb{Z}^+ \to \mathbb{Z}^+$ such that the following is true. Suppose $\mathcal{B}$ is a factor defined by polynomials $P_1, \dots, P_C : \mathbb{K}^n \to \mathbb{T}$ of additive degree at most d. Then, there is an r-regular factor $\mathcal{B}'$ consisting of polynomials $Q_1, \dots, Q_{C'} : \mathbb{K}^n \to \mathbb{T}$ of additive degree $\leqslant d$ such that $\mathcal{B}' \succeq_{sem} \mathcal{B}$ and $C' \leqslant C_{32}^{(r,d)}(C)$.*

Moreover, if $\mathcal{B}$ is itself a refinement of some polynomial factor $\hat{\mathcal{B}}$ that has rank $> (r(C') + C')$, then additionally $\mathcal{B}'$ will be a syntactic refinement of $\hat{\mathcal{B}}$.

Proof. Follows directly from Lemma 2.18 of [9] by identifying $\mathbb{K}^n$ with $\mathbb{F}^{rn}$. ◀

In fact, the regularization process of Lemma 32 can be implemented in time $O(n^{d+1})$ [12].

Finally, we'll use the Gowers inverse theorem proved by Tao and Ziegler [48] for non-classical polynomials mapping to $\mathbb{T}$.

▶ **Theorem 33** (Theorem 1.20 of [48]). *Suppose $\delta > 0$ and $d \geqslant 1$ is an integer. There exists an $r = r_{33}(\delta, d)$ such that the following holds. If a non-classical polynomial $P : \mathbb{K}^n \to \mathbb{T}$ with degree d satisfies $\|P\|_{U^d} \geqslant \delta$, then* $\operatorname{rank}(P) \leqslant r$.

4.2 Equidistribution results

Let us start with the following simple observation.

▶ **Lemma 34.** *Given $\varepsilon > 0$, let $\mathcal{B}$ be a polynomial factor of degree $d > 0$, complexity C and rank $r_{34}(d, \varepsilon)$, defined by a sequence of non-classical polynomials $P_1, \dots, P_C : \mathbb{K}^n \to \mathbb{T}$ having respective depths $k_1, \dots, k_C$. Suppose $\alpha = (\alpha_1, \dots, \alpha_C) \in \mathbb{U}_{k_1+1} \times \cdots \times \mathbb{U}_{k_C+1}$. Then:*

$$\mathbf{Pr}_x[\mathcal{B}(x) = \alpha] = \frac{1}{\|\mathcal{B}\|} \pm \varepsilon.$$

Proof. This is standard. See for example lemma 3.2 of [9]. ◀

In our applications though, we will often need not just $\mathcal{B}(x)$ to be nearly uniformly distributed but the tuple $(\mathcal{B}(x) : x \in H)$ for a set $H \subseteq \mathbb{K}^n$ to be nearly uniformly distributed. In particular, we consider the case when H is an affine subspace of $\mathbb{K}^n$. The following lemma is key.

▶ **Lemma 35** (Near orthogonality). *Let $A = (L_1, \dots, L_m)$ be a weight-closed affine constraint of bounded size on ℓ variables. Suppose $\mathcal{B}$ is a polynomial factor of degree d and rank $\geqslant r^{(33)}(d, \delta)$, defined by the sequence of non-classical polynomials $P_1, \dots, P_c : \mathbb{K}^n \to \mathbb{T}$. Let $\Lambda = (\lambda_{ij})_{i \in [c], j \in [m]}$ be a tuple of integers. Define:*

$$P_\Lambda(x_1, \dots, x_k) = \sum_{i \in [c], j \in [m]} \lambda_{ij} P_i(L_j(x_1, \dots, x_\ell)).$$

Then one of the following is true.

1. *For every $i \in [c]$, it holds that $\sum_{j \in [m]} \lambda_{ij} Q_i(L_j(\cdot)) \equiv 0$ for all polynomials $Q_i : \mathbb{K}^n \to \mathbb{T}$ with the same degree and depth as P_i. Clearly, this implies $P_\Lambda \equiv 0$.*
2. *$P_\Lambda \not\equiv 0$. Moreover,* $\operatorname{bias}(P_\Lambda) \leqslant \delta$.

Proof. For $j \in [m]$, let $(w_{j,1}, \dots, w_{j,\ell}) \in \mathbb{K}^\ell$ denote the affine form given by L_j. Note that $w_{j,1} = 1$.

For each i, we do the following. If for some j, we have $\mathsf{wt}(L_j) > \deg(\lambda_{i,j} P_i)$, $\lambda_{i,j} \neq 0$, then, $L_j(x_1, \dots, x_\ell) = x_1 + \sum_{i=2}^{\ell} \left(\sum_{k=1}^{r} u_{i,k} \cdot \beta_k\right) x_i$ where $\boldsymbol{\beta}$ is the dual basis to $\boldsymbol{\alpha}$, each $u_{i,k} \in [0, p-1]$ and $\sum_{i,k} u_{i,k} > \deg(\lambda_{i,j} P_i)$. Using Definition 2, we can replace $\lambda_{i,j} P_i(L_j)$ by a $\mathbb{Z}$-linear combination of $P_i(L_{j'})$ where $L_{j'} \preceq L_j$ until no such j exists. This is where we use the fact that the affine constraint is weight-closed. Suppose the new coefficients are denoted by $(\lambda'_{i,j})$. If the $\lambda'_{i,j}$ are all zero, then for every $i \in [c]$ individually, $\sum_{j \in [m]} P_i(L_j(x_1, \dots, x_\ell)) \equiv 0$. Indeed, $\sum_{j \in [m]} Q_i(L_j(x_1, \dots, x_\ell)) \equiv 0$ for any Q_i with the same degree and depth, as the transformation from $\lambda_{i,j}$ to $\lambda'_{i,j}$ did not use any other information about P_i.

Else some $\lambda'_{i,j} \neq 0$. Also, $\mathsf{wt}_{\boldsymbol{\alpha}}(L_j) \leqslant \deg(\lambda'_{i,j} P_i)$. Then we show the second part of the lemma, that is $|\mathbb{E}[e(P_\Lambda(x_1, \dots, x_k)]| \leqslant \delta$.

Suppose without loss of generality that the following is true.

- $\lambda'_{i,1} \neq 0$ for some $i \in [C]$.
- L_1 is maximal in the sense that for every $j \neq 1$, either $\lambda'_{i,j} = 0$ for all $i \in [C]$ or $\mathsf{wt}_{\boldsymbol{\alpha}}(w_{j,s}) < \mathsf{wt}_{\boldsymbol{\alpha}}(w_{1,s})$ for some $s \in [\ell]$.

For $a = (a_1, \dots, a_\ell) \in \mathbb{K}^\ell$ and $y \in \mathbb{K}^n$ and $P : \mathbb{K}^n \to \mathbb{T}$, define

$$\overline{D}_{a,y}P(x_1, \dots, x_\ell) = P(x_1 + a_1 y, \dots, x_\ell + a_\ell y) - P(x_1, \dots, x_\ell).$$

Then

$$\overline{D}_{a,y}(P_i \circ L_j)(x_1, \dots, x_\ell) = (D_{L_j(a)y}P_i)(L_j(x_1, \dots, x_\ell)).$$

Let $\Delta = \mathsf{wt}_\alpha(L_1) \leqslant d$. Define $a_1, \dots, a_\Delta$ be the set of vectors of the form $(-w, 0, \dots, 1, 0, \dots, 0)$ where 1 is in the ith coordinate for $i \in [2, \ell]$ and for all $w \in \mathbb{K}$ satisfying $0 \leqslant \mathsf{wt}_{\boldsymbol{\alpha}}(w) < \mathsf{wt}_{\boldsymbol{\alpha}}(w_{1,i})$. Note that $\langle L_1, a_k \rangle \neq 0$ for $k \in [\Delta]$ but for any $j > 1$ there exists some $k \in [\Delta]$ such that $\langle L_j, a_k \rangle = 0$. Thus,

$$\mathop{\mathbb{E}}_{y_1,\dots,y_\Delta,x_1,\dots,x_\ell} \left[e\left((\overline{D}_{a_\Delta, y_\Delta} \dots \overline{D}_{a_1 . y_1} P_\Lambda)(x_1, \dots, x_\ell) \right) \right] = \left\| \sum_{i=1}^{C} \lambda'_{i,1} P_i \right\|_{U^\Delta}^{2^\Delta}.$$

The rest of the analysis is same as Theorem 3.3 in [9] and we skip it here. ◀

We can now use Lemma 35 to prove our result on equidistribution of regular factors over affine subspaces of $\mathbb{K}^n$.

▶ **Theorem 36.** *Let $\varepsilon > 0$. Let $\mathcal{B}$ be a polynomial factor defined by non-classical polynomials $P_1, \dots, P_c : \mathbb{K}^n \to \mathbb{T}$ with respective degrees $d_1, \dots, d_c \in \mathbb{Z}^+$ and depths $k_1, \dots, k_c \in \mathbb{Z}^{\geqslant 0}$. Suppose $\mathcal{B}$ has rank at least $r^{(33)}(d, \varepsilon)$ where $d = \max(d_1, \dots, d_c)$. Let $A = (L_1, \dots, L_m)$ be a weight-closed affine constraint. For every $i \in [c]$, define Λ_i to be the set of tuples $(\lambda_1, \dots, \lambda_m) \in [0, p^{k_i+1} - 1]$ such that $\sum_{j=1}^m \lambda_j Q_i(L_j(\cdot)) \equiv 0$ for all non-classical polynomials Q_i with the same degree and depth as P_i.*

Consider $(\alpha_{i,j} : i \in [c], j \in [m]) \in \mathbb{T}^{cm}$ such that for every $i \in [c]$ and for every $(\lambda_1, \dots, \lambda_m) \in \Lambda_i$, $\sum_{j=1}^m \lambda_j \alpha_{i,j} = 0$. Then:

$$\mathbf{Pr}_{x_1,\dots,x_\ell \in \mathbb{K}^n}[\mathcal{B}(L_j(x_1, \dots, x_\ell)) = (\alpha_{1,j}, \dots, \alpha_{c,j})\ \forall j \in [m]] = \frac{\prod_{i=1}^c |\Lambda_i|}{\|\mathcal{B}\|^m} \pm \varepsilon$$

Proof.

$$\begin{aligned}
&\mathbf{Pr}_{x_1,\dots,x_\ell \in \mathbb{K}^n}[\mathcal{B}(L_j(x_1, \dots, x_\ell)) = (\alpha_{1,j}, \dots, \alpha_{c,j})\ \forall j \in [m]] \\
&= \mathop{\mathbb{E}}_{x_1,\dots,x_\ell} \left[\prod_{i,j} \frac{1}{p^{k_i+1}} \sum_{\lambda_{i,j}=0}^{p^{k_i+1}-1} e(\lambda_{i,j}(P_i(L_j(x_1, \dots, x_\ell)) - \alpha_{i,j})) \right] \\
&= \left(\prod_i p^{-(k_i+1)} \right)^m \sum_{\substack{(\lambda_{i,j}) \\ \in \prod_{i,j}[0, p^{k_i+1}-1]}} e\left(-\sum_{i,j} \lambda_{i,j}\alpha_{i,j} \right) \mathbb{E}\left[e\left(\sum_{i,j} \lambda_{i,j} P_i(L_j(x_1, \dots, x_\ell)) \right) \right] \\
&= p^{-m \sum_{i=1}^c (k_i+1)} \cdot \left(\prod_{i=1}^c |\Lambda_i| \pm \varepsilon p^{m \sum_{i=1}^c (k_i+1)} \right)
\end{aligned}$$

The last line is due to the observation that from Lemma 35, $\sum_{i=1}^c \sum_{j=1}^m \lambda_{i,j} P_i(L_j(x_1, \dots, x_\ell)) \equiv 0$ if and only if for every $i \in [c]$, $(\lambda_{i,1}, \dots, \lambda_{i,m}) \in \Lambda_i \pmod{p^{k_i+1}}$. So, $\sum_{i,j} \lambda_{i,j} P_i(L_j(\cdot))$ is identically 0 for $\prod_i |\Lambda_i|$ many tuples $(\lambda_{i,j})$ and for those tupes, $\sum_{i,j} \lambda_{i,j}\alpha_{i,j} = 0$ also. ◀

Note that in Theorem 36, if ε is a constant, m needs to be bounded for the claim to be non-trivial, which in turn requires that the affine forms in L be of bounded weight.

4.3 Preservation of Locally Characterized Properties

4.3.1 Local Characterization

As described in the introduction, by a locally characterized property, we informally mean a property for which non-membership can be certified by a finite sized witness. Specifically for affine-invariant properties, we define:

▶ **Definition 37** (Locally characterized properties).

- An *induced affine constraint of size m on ℓ variables* is a pair (A, σ) where A is an affine constraint of size m on ℓ variables and $\sigma \in [R]^m$.
- Given such an induced affine constraint (A, σ), a function $f : \mathbb{K}^n \to [R]$ is said to be (A, σ)-*free* if there exist no $x_1, \dots, x_\ell \in \mathbb{K}^n$ such that $(f(L_1(x_1, \dots, x_\ell)), \dots, f(L_m(x_1, \dots, x_\ell))) = \sigma$. On the other hand, if such $x_1, \dots, x_\ell$ exist, we say that *f induces (A, σ) at $x_1, \dots, x_\ell$*.
- Given a (possibly infinite) collection $\mathcal{A} = \{(A^1, \sigma^1), (A^2, \sigma^2), \dots, (A^i, \sigma^i), \dots\}$ of induced affine constraints, a function $f : \mathbb{K}^n \to [R]$ is said to be *$\mathcal{A}$-free* if it is (A^i, σ^i)-free for every $i \geqslant 1$. The size of $\mathcal{A}$ is the size of the largest induced affine constraint in $\mathcal{A}$.
- Additionally, $\mathcal{A} = \{(A^1, \sigma^1), (A^2, \sigma^2), \dots, (A^K, \sigma^K)\}$ is a W-light affine system if there exists a basis $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_r)$ such that $\mathsf{wt}_{\boldsymbol{\alpha}}(A^i) \leqslant W$ for all $i \in [K]$.
- A property $\mathcal{P} \subseteq \{\mathbb{K}^n \to [R]\}$ is said to be *K, W-lightly locally characterized* if it is equivalent to $\mathcal{A}$-freeness for some W-light affine system $\mathcal{A}$ whose size is $\leqslant K$.

We recall that Kaufman and Ron [38] show that:

▶ **Theorem 38** ([38]). *The property $\mathcal{P}_d = \{P : \mathbb{K}^n \to \mathbb{K} : \deg(P) \leqslant d\}$ is $q^{\lceil (d+1)/(q-q/p) \rceil}$, pr $\lceil (d+1)/(q-q/p) \rceil$-lightly locally characterized.*

4.3.2 Main Result on Property Preservation

▶ **Theorem 39.** *Let $\mathcal{P} \subset \{\mathbb{K}^n \to \mathbb{K}\}$ be a K, W-lightly locally characterized property. For an integer d, suppose $P_1, \dots, P_c : \mathbb{K}^n \to \mathbb{T}$ are polynomials of additive degree $\leqslant d$, forming a factor of rank $> r_{39}(d, K)$, and $\Gamma : \mathbb{T}^c \to \mathbb{K}$ is a function such that $F : \mathbb{K}^n \to \mathbb{K}$ defined by $F(x) = \Gamma(P_1(x), \dots, P_c(x))$ satisfies $\mathcal{P}$.*

For every collection of additive polynomials $Q_1, \dots, Q_c : \mathbb{K}^n \to \mathbb{T}$ with $\deg(Q_i) \leqslant \deg(P_i)$ and $\operatorname{depth}(Q_i) \leqslant \operatorname{depth}(P_i)$ for all $i \in [c]$, if $G : \mathbb{K}^n \to \mathbb{K}$ is defined by $G(x) = \Gamma(Q_1(x), \dots, Q_c(x))$, then $G \in \mathcal{P}$ too.

Proof. For the sake of contradiction, suppose $G \notin \mathcal{P}$. Then, for a weight-closed affine constraint consisting of K' linear forms $L_1, \dots, L_{K'}$, there exist $x_1, \dots, x_\ell$ such that $(G(L_1(x_1, \dots, x_\ell)), \dots, G(L_{K'}(x_1, \dots, x_\ell)))$ which form a witness to $G \notin \mathcal{P}$. Note that K' is a function of only K and W because the affine forms characterizing $\mathcal{P}$ can be made weight $\leqslant W$ by a choice of basis for $\mathbb{K}$ over $\mathbb{F}$ and then completed into a weight-closed constraint. So, there exists $x_1, \dots, x_\ell \in \mathbb{K}^n$ such that the tuple $B = (Q_i(L_j(x_1, \dots, x_\ell)) : j \in [K'], i \in [c]) \in \mathbb{T}^{cK'}$ is a proof of the fact that $G \notin \mathcal{P}$.

Now we argue that there exist $x'_1, \dots, x'_\ell$ such that $(P_i(L_j(x'_1, \dots, x'_\ell)) : i \in [c], j \in [K])$ equals B, thus showing that $F \notin P$, a contradiction. Notice that B satisfies the conditions required of α in Theorem 36. So by Theorem 36,

$$\mathbf{Pr}_{x'_1, \dots, x'_\ell} [(P_i(L_j(x'_1, \dots, x'_\ell) : i \in [c], j \in [K]) = B] > 0$$

if the rank of the factor formed by $P_1, \ldots, P_c$ is more than $r^{(33)}\left(d, \frac{1}{2\|\mathcal{B}\|^K}\right)$, where $\|\mathcal{B}\| = p^{\sum_{i=1}^{c}(\mathrm{depth}(P_i)+1)}$. ◀

In our applications, we will use Theorem 39 for the property of having bounded degree, which is lightly locally characterized by Theorem 38.

5 List decoding of RM codes

We state the following corollary which we need in the proof to follow. We only state a special case of it which is enough.

▶ **Corollary 40** (Corollary 3.3 of [14]). *Let $g : K \to K$, $\varepsilon > 0$. Then there exist $c \leqslant 1/\varepsilon^2$ functions $h_1, h_2, \ldots, h_c \in \mathrm{RM}_{\mathbb{K}}(n, d)$ such that for every $f \in \mathrm{RM}_{\mathbb{K}}(n, d)$, there is a function $\Gamma_f : \mathbb{K}^c \to \mathbb{K}$ such that*

$$\mathbf{Pr}_x[\Gamma_f(h_1(x), \ldots, h_c(x)) = f(x)] \geqslant \mathbf{Pr}_x[g(x) = f(x)] - \varepsilon.$$

▶ **Theorem 2** (restated). *Let $\mathbb{K} = \mathbb{F}_q$ be an arbitrary finite field. Let $\varepsilon > 0$ and $d, n \in \mathbb{N}$. Then,*

$$\ell_{\mathbb{K}}(d, n, \delta_{\mathbb{K}}(d) - \varepsilon) \leqslant c_{q,d,\varepsilon}.$$

Proof. We follow the proof structure in [14]. Let $g : \mathbb{K}^n \to \mathbb{K}$ be a received word. Apply Corollary 40 with approximation parameter $\varepsilon/2$ gives $\mathcal{H}_0 = \{h_1, \ldots, h_c\} \subseteq \mathrm{RM}_{\mathbb{K}}(n, d)$, $c \leqslant 4/\varepsilon^2$ such that, for every $f \in \mathrm{RM}_{\mathbb{K}}(n, d)$, there is a function $\Gamma_f : \mathbb{K}^c \to \mathbb{K}$ satisfying

$$\mathbf{Pr}[\Gamma_f(h_1(x), h_2(x), \ldots, h_c(x)) = f(x)] \geqslant \mathbf{Pr}[g(x) = f(x)] - \varepsilon/2.$$

B

$$\mathbf{Pr}[\Gamma'_f(\mathrm{Tr}(\alpha_i h_j(x)) : 1 \leqslant i \leqslant r, 1 \leqslant j \leqslant c) = F(\mathrm{Tr}(\alpha_i f(x)) : 1 \leqslant i \leqslant r)] \geqslant d/q + \varepsilon/2,$$

where $\Gamma'_f : \mathbb{F} \to \mathbb{K}$ and $F : \mathbb{F}^r \to \mathbb{K}$. From here onwards, we identify $\mathbb{F}$ with $\mathbb{U}_1$. Let $\mathcal{H} = \{\mathrm{Tr}(\alpha_i h_j(x)) : 1 \leqslant i \leqslant r, 1 \leqslant j \leqslant c\}$ and $\mathcal{H}_F = \{\mathrm{Tr}(\alpha_i f(x)) : 1 \leqslant i \leqslant r)\}$.

Let $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two non decreasing functions to be specified later, and let $C^{(32)}_{r,d}$ be as given in Lemma 32. We will require that for all $m \geqslant 1$,

$$r_1(m) \geqslant r_2(C^{(32)}_{r_2,d}(m+1)) + C^{(32)}_{r_2,d}(m+1) + 1. \tag{6}$$

As a first step, we r_1-regularize $\mathcal{H}$ by Lemma 32. This gives an r_1-regular factor $\mathcal{B}'$ of degree at most d, defined by polynomials $H_1, \ldots, H_c : \mathbb{K}^n \to \mathbb{T}$, $c' \leqslant C^{(32)}_{r_1,d}(cr)$ and $\mathrm{rank}(\mathcal{B}') \geqslant r_1(c')$. We denote $\mathcal{H}' = \{H_1, \ldots, H_{c'}\}$. Let $\mathrm{depth}(H_i) = k_i$ for $i \in [c']$. Let $G_f : \otimes_{i=1}^{c'} \mathbb{U}_{k_i+1} \to \mathbb{U}_1$ be defined such that

$$\Gamma_f(h_1(x), \ldots, h_c(x)) = G_f(h'_1(x), \ldots, h'_{c'}(x)).$$

Next, given any polynomial $f : \mathbb{K}^n \to \mathbb{K}$ of degree at most d, we will show that if $\mathbf{Pr}[f(x) \neq g(x)] \leqslant \delta(d) - \varepsilon$, then f is measurable with respect to $\mathcal{H}'$ and this would upper bound the number of such polynomials by $c'(q, d, \varepsilon)$ independent on n.

Fix such a polynomial f. Call $F_i = \mathrm{Tr}(\alpha_i f)$. Appealing again to Lemma 32, we r_2-regularize $\mathcal{B}_f := \mathcal{B}' \bigcup \mathcal{H}_F$. We get an r_2-regular factor $\mathcal{B}'' \succeq_{syn} \mathcal{B}'$ defined by the collection

$\mathcal{H}'' = \{H_1, \dots, H_{c'}, H'_1, \dots, H'_{c''}\}$. Note that it is a syntactic refinement of $\mathcal{B}'$ as by our choice of r_1,

$$\operatorname{rank}(\mathcal{B}') \geqslant r_1(c') \geqslant r_2(C^{(32)}_{r_2,d}(c'+1)) + C^{(32)}_{r_2,d}(c'+1) + 1 \geqslant r_2(|\mathcal{B}''|) + |\mathcal{B}''| + 1.$$

We will choose r_2 such that for all $m \geqslant 1$,

$$r_2(m) = \max\left(r_d^{(34)}\left(\frac{\varepsilon/4}{\left(p^{\lfloor \frac{d-1}{p-1} \rfloor + 1}\right)^m} \right), r_d^{(39)}(m) \right). \tag{7}$$

Since each F_i is measurable with respect to $\mathcal{B}''$, there exists $F' : S \to \mathbb{U}_1$ such that

$$f(x) = F'(H_1(x), \dots, H_{c'}(x), H'_1(x), \dots, H'_{c''}(x)).$$

Summing up, we have

$$\mathbf{Pr}[G(H_1(x), H_2(x), \dots, H_{c'}(x)) = F'(H_1(x), \dots, H_{c'}(x), H'_1(x), \dots, H'_{c''}(x))] \geqslant d/q + \varepsilon/2.$$

We next show that we can have each polynomial in the factor have a disjoint set of inputs. This would simplify the analysis considerably.

► **Claim 41.** *Let x^i, y^j, $i \in [c'], j \in [c'']$ be pairwise disjoint sets of $n \in \mathbb{N}$ variables each. Let $n' = n(c' + c'')$. Let $\tilde{f} : \mathbb{K}^{n'} \to \mathbb{K}$ and $\tilde{g} : \mathbb{K}^{n'} \to \mathbb{K}$ be defined as*

$$\tilde{f}(x) = F(H_1(x^1), \dots, H_{c'}(x^{c'}), H'_1(y^1), \dots, H'_{c''}(y^{c''}))$$

and

$$\tilde{g}(x) = G(H'_1(x^1), \dots, H_{c'}(x^{c'})).$$

Then $\deg(\tilde{f}) \leqslant d$ *and*

$$\left| \mathbf{Pr}_{x \in \mathbb{F}^{n'}}[\tilde{f}(x) = \tilde{g}(x)] - \mathbf{Pr}_{x \in \mathbb{F}^n}[f(x) = G_f(h'_1(x), h'_2(x), \dots, h'_c(x))] \right| \leqslant \varepsilon/4.$$

Proof. The bound $\deg(\tilde{f}) \leqslant \deg(f) \leqslant d$ follows from Lemma 39. To establish the bound on $\mathbf{Pr}[\tilde{f} = \tilde{g}]$, for each $s \in S$ let

$$p_1(s) = \mathbf{Pr}_{x \in \mathbb{F}^n}[(h'_1(x), \dots, h'_{c'}(x), h''_1(x), \dots, h''_{c''}(x)) = s].$$

Applying Lemma 34 and since our choice of r_2 satisfies $\operatorname{rank}(\mathcal{H}'') \geqslant r_d^{(34)}(\varepsilon/4|S|)$, we have that p_1 is nearly uniform over S,

$$p_1(s) = \frac{1 \pm \varepsilon/4}{|S|}.$$

Similarly, let

$$p_2(s) = \mathbf{Pr}_{x^1, \dots, x^{c'}, y^1, \dots, y^{c''} \in \mathbb{F}^n}[(h'_1(x^1), \dots, h'_{c'}(x^{c'}), h''_1(y^1), \dots, h''_{c''}(y^{c''})) = s].$$

Note that the rank of the collection of polynomials $\{h'_1(x^1), \dots, h'_{c'}(x^{c'}), h''_1(y^1), \dots, h''_{c''}(y^{c''})\}$ defined over $\mathbb{F}^{n'}$ cannot be lower than that of $\mathcal{H}''$. Applying Lemma 34 again gives

$$p_2(s) = \frac{1 \pm \varepsilon/4}{|S|}.$$

For $s \in S$, let $s' \in \otimes_{i=1}^{c'} \mathbb{U}_{k_i+1}$ be the restriction of s to first c' coordinates, that is, $s' = (s_1, \dots, s_{c'})$. Thus

$$\begin{aligned}\mathbf{Pr}_{x\in\mathbb{F}^{n'}}[\tilde{f}(x) = \tilde{g}(x)] &= \sum_{s\in S} p_2(s) 1_{F(s)=G_f(s')}\\ &= \sum_{s\in S} p_1(s) 1_{F(s)=G_f(s')} \pm \varepsilon/4\\ &= \mathbf{Pr}_{x\in\mathbb{F}^n}[f(x) = G_f(h_1'(x), h_2'(x), \dots, h_c'(x))] \pm \varepsilon/4.\end{aligned}$$

◀

So, we obtain that

$$\mathbf{Pr}_{x\in\mathbb{F}^{n'}}[\tilde{f}(x) = \tilde{g}(x)] \geqslant \mathbf{Pr}_{x\in\mathbb{F}^n}[f(x) = G_f(h_1'(x), \dots, h_{c'}'(x))] - \varepsilon/4 \geqslant 1 - \delta(d) + \varepsilon/4.$$

Next, we need the following variant of the Schwartz-Zippel lemma from [14].

▶ **Claim 42.** *Let $d, n_1, n_2 \in \mathbb{N}$. Let $f_1 : \mathbb{K}^{n_1+n_2} \to \mathbb{K}$ and $f_2 : \mathbb{K}^{n_1} \to \mathbb{K}$ be such that $\deg(f_1) \leqslant d$ and*

$$\mathbf{Pr}[f_1(x_1, \dots, x_{n_1+n_2}) = f_2(x_1, \dots, x_{n_1})] > 1 - \delta(d)$$

Then, f_1 does not depend on $x_{n_1+1}, \dots, x_{n_1+n_2}$.

With Claim 42 applied to $f_1 = \tilde{f}, f_2 = \tilde{g}, n_1 = nc', n_2 = nc''$. We obtain that $\tilde{f}$ does not depend on $y^1, \dots, y^{c''}$. Hence,

$$\tilde{f}(x^1, \dots, x^{c'}, y^1, \dots, y^{c''}) = F(H_1'(x^1), \dots, H_{c'}'(x^{c'}), C_1, \dots, C_{c''})$$

where $C_j = H_j''(0)$ for $j \in [c'']$. If we substitute $x^1 = \dots = x^{c'} = x$ we get that

$$f(x) = F(H_1'(x), \dots, H_{c'}'(x), H_1''(x), \dots, H_{c''}''(x)) = F(H_1'(x), \dots, H_{c'}'(x), C_1, \dots, C_{c''}),$$

which shows that f is measurable with respect to $\mathcal{H}'$, as claimed. ◀

6 Polynomial decomposition

We first formally define the problem for which we claim a polynomial time algorithm.

▶ **Definition 43.** Given $k \in \mathbb{N}$ and $\Delta = (\Delta_1, \dots, \Delta_k) \in \mathbb{N}^k$ and a function $\Gamma : \mathbb{K}^k \to \mathbb{K}$, a function $P : \mathbb{K}^n \to \mathbb{K}$ is (k, Δ, Γ)-structured if there exist polynomials $P_1, \dots, P_k : \mathbb{K}^n \to \mathbb{K}$ with $\deg(P_i) \leqslant \Delta_i$ such that for $x \in \mathbb{K}^n$, we have

$$P(x) = \Gamma(P_1(x), \dots, P_k(x)).$$

The polynomials $P_1, \dots, P_k$ form a (k, Δ, Γ)-decomposition.

The main result we prove is the following.

▶ **Theorem 44.** *Let $k \in \mathbb{N}$. For every $\Delta = (\Delta_1, \dots, \Delta_k) \in \mathbb{N}^k$ and every function $\Gamma : \mathbb{K}^k \to \mathbb{K}$, there is a randomized algorithm A that on input $P : \mathbb{K}^n \to \mathbb{K}$ of degree d, runs in time $\mathrm{poly}_{q,k,\Delta}(n^{d+1})$ and outputs a (k, Δ, Γ)-decomposition of P if one exists while otherwise returning NO.*

We first show that the notion of rank is robust to hyperplane restrictions over nonprime fields. More precisely, we have the following.

▶ **Lemma 45.** *Let $P : \mathbb{K}^n \to \mathbb{T}$ be a non-classical polynomial such that* $\operatorname{rank}(P) \geqslant r$*. Let H be a hyperplane in $\mathbb{K}^n$. Then the restriction of P to H has rank at least $r - q$.*

Proof. Without loss of generality, let H be defined by $x_1 = 0$. Let $P' : \mathbb{K}^{n-1} \to \mathbb{T}$ be the restriction of P defined by $P'(y) = P(0y)$. Let $\pi : \mathbb{K}^n \to \mathbb{K}^{n-1}$ be the map $\pi(x_1x_2 \dots x_n) = x_2 \dots x_n$. Let $P'' : \mathbb{K}^n \to \mathbb{T}$ be defined by $P''(x) = P(x) - P' \circ \pi$. Then $P''(x) = 0$ for $x \in H$. For $i \in \mathbb{K} \setminus \{0\}$, let $h_i = (i, 0, \dots, 0)$. Then, for $y \in H$, define $R_j : \mathbb{K}^n \to \mathbb{T}$ by

$$R_j(y) = P''(y + h_j) = (D_{h_j}P'')(y).$$

Note that $\deg(R_j) \leqslant d - 1$. Now, since $P(x) = P''(x) + P' \circ \pi(x)$, we have

$$P(x) = \Gamma(P' \circ \pi, x_1, \{R_y(x) : y \in \mathbb{F}\}).$$

Now, if $\operatorname{rank}(P') \leqslant r$, then $\operatorname{rank}(P' \circ \pi) \leqslant r$ and hence $\operatorname{rank}(P) \leqslant r + q$. This finishes the proof. ◀

We now start with the proof of Theorem 44.

Proof. Let $R_1 : \mathbb{N} \to \mathbb{N}$ be defined as $R_1(m) = R_2(c_{32}^{(R_1,d)}(m+k)) + c_{32}^{(R_1,d)}(m+k) + q$ where $R_2 : \mathbb{N} \to \mathbb{N}$ will be specified later.

We have that $P(x) = \sum_i \beta_i \operatorname{Tr}(\alpha_i P(x))$ for the dual basis $\beta_1, \dots, \beta_r$. Set $f_i(x) = \operatorname{Tr}(\alpha_i P(x))$. Identifying $\mathbb{F}$ with $\mathbb{U}_1$ we treat $f_i : \mathbb{K}^n \to \mathbb{T}$. Regularize $\{f_1, \dots, f_r\}$ using the algorithm of [12] to find R_1-regular $\mathcal{B} = \{g_1, \dots, g_C : \mathbb{K}^n \to \mathbb{T}\}$ where $C \leqslant c_{32}^{(R_1,d)}(r)$. So, $f_i(x) = G_i(g_1(x), \dots, g_C(x))$ and $P(x) = \sum_i \alpha_i G_i(g_1(x), \dots, g_C(x))$. Thus, if $n \leqslant Cd$, then we are done by a brute force search.

Else, $n > Cd$. For each g_i, pick a monomial m_i with degree $\deg(P_i)$. Then there is $i_0 \in [n]$ such that x_{i_0} does not appear in any g_i. Set $g_i' := g_i|x_{i_0} = 0$. Let $\mathcal{B}'$ be the factor defined by the $g_i's$. Note that $\deg(g_i') = \deg(g_i)$ and $\operatorname{depth}(g_i') = \operatorname{depth}(g_i)$. Also, by Lemma 45, $\mathcal{B}'$ is $R_1 - q$-regular.

Now, using recursion, we solve the problem on $n - 1$ variables. That is, decide if for $P' := P|x_{i_0} = 0$ is (k, Δ, Γ)-structured. If P' is not, then P is not either, so we are done. Else, suppose the algorithm does not output NO.

Say

$$P'(x) = \Gamma(S_1(x), \dots, S_k(x)) = \Gamma'(\operatorname{Tr}(\alpha_j S_i(x)) : i \in [k], j \in [r]),$$

where

$$\Gamma'(a_{ij} : i \in [k], j \in [r]) = \Gamma(\sum_j \alpha_i a_{ij} : i \in [k]).$$

Note that while $\Gamma : \mathbb{K}^k \to \mathbb{K}$, we have $\Gamma' : \mathbb{F}^{kr} \to \mathbb{K}$. Let $\mathcal{B}_1$ be the factor formed by $\{\operatorname{Tr}(\alpha_j S_i)\}$. Via the algorithm of [12], regularize $\mathcal{B}' \cup \mathcal{B}_1$ using $R_2 : \mathbb{N} \to \mathbb{N}$ and we get a syntactic refinement $\mathcal{B}' \cup \mathcal{B}_1'$ by the choice of R_1. Let $\mathcal{B}_1' = \{s_1', \dots, s_D'\}$. where

$$\operatorname{Tr}(\alpha_j S_i) = G_{ij}(g_i', s_j' : i \in [C], j \in [D]).$$

Choose R_2 large enough such that the map induced by $\mathcal{B}' \cup \mathcal{B}'_1$ is surjective. Now, fix any $\ell \in [r]$. Then,

$$\mathrm{Tr}(\alpha_\ell P') = G_\ell(g'_1, \ldots, g'_C) = F_\ell(G_{ij}(g'_i, s'_j)),$$

where $F_\ell = \mathrm{Tr}(\alpha_\ell \Gamma')$. Thus, for $a_1, \ldots, a_C, b_1, \ldots b_D \in \mathbb{F}$,

$$G_\ell(a_1, \ldots, a_C) = F_\ell(G_{ij}(a_1, \ldots, b_D) : i \in [C], j \in [D]).$$

Substituting, $a_i = g_i(x)$ and $b_j = 0$ we have

$$\mathrm{Tr}(\alpha_\ell P) = G_\ell(g_1, \ldots, g_C) = F_\ell(G_{ij}(g_i, 0)).$$

Now,

$$\mathrm{Tr}(\alpha_\ell P) = \mathrm{Tr}(\alpha_\ell \Gamma(Q_i : i \in [k])),$$

where $Q_i(x) = \sum_{j=1}^{r} \alpha_j G_{ij}(g'_i, \ldots, 0)$.

Since, this is true for all $\ell \in [r]$, we have

$$P(x) = \Gamma(Q_1(x), \ldots, Q_k(x)).$$

where Q_i is defined as above. This finishes the proof. ◀

References

1 Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing Reed-Muller codes. *IEEE Trans. Inform. Theory*, 51(11):4032–4039, 2005. `doi:10.1109/TIT.2005.856958`.

2 S. Arora and M. Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

3 László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd Annual ACM Symposium on the Theory of Computing*, pages 21–32, New York, 1991. ACM Press.

4 László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

5 Eli Ben-Sasson, Ghid Maatouk, Amir Shpilka, and Madhu Sudan. Symmetric LDPC codes are not necessarily locally testable. In *Proc. 26th Annual Conference on Computational Complexity (CCC)*, pages 55–65. IEEE, 2011.

6 Arnab Bhattacharyya. Polynomial decompositions in polynomial time. In *Proc. 22nd Annual European Symposium on Algorithms*, pages 125–136, 2014.

7 Arnab Bhattacharyya and Abhishek Bhowmick. Using higher-order fourier analysis over general fields. *CoRR*, abs/1505.00619, 2015. URL: `http://arxiv.org/abs/1505.00619`.

8 Arnab Bhattacharyya, Victor Chen, Madhu Sudan, and Ning Xie. Testing linear-invariant non-linear properties. *Theory Comput.*, 7(1):75–99, 2011.

9 Arnab Bhattacharyya, Eldar Fischer, Hamed Hatami, Pooya Hatami, and Shachar Lovett. Every locally characterized affine-invariant property is testable. In *Proc. 45th Annual ACM Symposium on the Theory of Computing*, pages 429–436, 2013.

10 Arnab Bhattacharyya, Eldar Fischer, and Shachar Lovett. Testing low complexity affine-invariant properties. In *Proc. 24th ACM-SIAM Symposium on Discrete Algorithms*, pages 1337–1355, 2013.

11 Arnab Bhattacharyya, Elena Grigorescu, and Asaf Shapira. A unified framework for testing linear-invariant properties. In *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 478–487, 2010.

12 Arnab Bhattacharyya, Pooya Hatami, and Madhur Tulsiani. Algorithmic regularity for polynomials and applications. In *Proc. 26th ACM-SIAM Symposium on Discrete Algorithms*, pages 1870–1889, 2015.

13 Abhishek Bhowmick and Shachar Lovett. Bias vs structure of polynomials in large fields, and applications in effective algebraic geometry and coding theory. *CoRR*, abs/1506.02047, 2015. URL: `http://arxiv.org/abs/1506.02047`.

14 Abhishek Bhowmick and Shachar Lovett. List decoding Reed-Muller codes over small fields. In *Proc. 47th Annual ACM Symposium on the Theory of Computing*, pages 277–285, New York, NY, USA, 2015. ACM.

15 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comp. Sys. Sci.*, 47:549–595, 1993. Earlier version in STOC'90.

16 A. Bogdanov and E. Viola. Pseudorandom bits for polynomials. In *Proc.* 48^{th} *IEEE Symp. on Foundations of Computer Science (FOCS'07)*, 2007.

17 Pierre Deligne. Application de la formule des traces aux sommes trigonometriques. In *SGA* $4\frac{1}{2}$ *Springer Lecture Notes in Matematics*, volume 569. Springer, 1978.

18 P. Elias. List decoding for noisy channels. Technical Report 335, Research Laboratory of Electronics, MIT, 1957.

19 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996.

20 O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proc.* 21^{st} *ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

21 O. Goldreich, R. Rubinfeld, and M. Sudan. Learning polynomials with queries: The highly noisy case. *SIAM J. Discrete Math.*, 13(4):535–570, 2000.

22 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45:653–750, 1998.

23 Oded Goldreich and Tali Kaufman. Proximity oblivious testing and the role of invariances. In *Studies in Complexity and Cryptography*, pages 173–190. Springer, 2011.

24 Oded Goldreich and Dana Ron. On proximity oblivious testing. *SIAM J. Comput.*, 40(2):534–566, 2011.

25 P. Gopalan. A Fourier-analytic approach to Reed-Muller decoding. In *Proc.* 51^{st} *IEEE Symp. on Foundations of Computer Science (FOCS'10)*, pages 685–694, 2010.

26 P. Gopalan, A. Klivans, and D. Zuckerman. List decoding Reed-Muller codes over small fields. In *Proc.* 40^{th} *ACM Symposium on the Theory of Computing (STOC'08)*, pages 265–274, 2008.

27 Parikshit Gopalan, Ryan O'Donnell, Rocco A. Servedio, Amir Shpilka, and Karl Wimmer. Testing Fourier dimensionality and sparsity. In *Proc. 36th Annual International Conference on Automata, Languages, and Programming*, pages 500–512, 2009.

28 William T. Gowers. A new proof of Szeméredi's theorem for arithmetic progressions of length four. *Geom. Funct. Anal.*, 8(3):529–551, 1998.

29 William T. Gowers. A new proof of Szeméredi's theorem. *Geom. Funct. Anal.*, 11(3):465–588, 2001.

30 Ben Green and Terence Tao. The distribution of polynomials over finite fields, with applications to the Gowers norms. *Contrib. Discrete Math.*, 4(2), 2009.

31 Elena Grigorescu, Tali Kaufman, and Madhu Sudan. Succinct representation of codes with applications to testing. *SIAM Journal on Discrete Mathematics*, 26(4):1618–1634, 2012.

32 Alan Guo, Swastik Kopparty, and Madhu Sudan. New affine-invariant codes from lifting. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 529–540. ACM, 2013.

33 V. Guruswami. *List Decoding of Error-Correcting Codes*, volume 3282 of *Lecture Notes in Computer Science*. Springer, 2004.
34 V. Guruswami. *Algorithmic Results in List Decoding*, volume 2 of *Foundations and Trends in Theoretical Computer Science*. Now Publishers, 2006.
35 Hamed Hatami and Shachar Lovett. Estimating the distance from testable affine-invariant properties. In *Proc. 54th Annual IEEE Symposium on Foundations of Computer Science*, pages 237–242. IEEE, 2013.
36 Tali Kaufman and Simon Litsyn. Almost orthogonal linear codes are locally testable. In *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 317–326. IEEE, 2005.
37 Tali Kaufman and Shachar Lovett. Worst case to average case reductions for polynomials. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 166–175, 2008.
38 Tali Kaufman and Dana Ron. Testing polynomials over general fields. *SIAM J. on Comput.*, 36(3):779–802, 2006.
39 Tali Kaufman and Madhu Sudan. Algebraic property testing: the role of invariance. In *Proc. 40th Annual ACM Symposium on the Theory of Computing*, pages 403–412, 2008.
40 Shachar Lovett, Roy Meshulam, and Alex Samorodnitsky. Inverse conjecture for the Gowers norm is false. In *Proc. 40th Annual ACM Symposium on the Theory of Computing*, pages 547–556, New York, NY, USA, 2008. ACM.
41 R. Pellikaan and X. Wu. List decoding of q-ary Reed-Muller codes. *IEEE Transactions on Information Theory*, 50(4):679–682, 2004.
42 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. on Comput.*, 25:252–271, 1996.
43 M. Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997. URL: `citeseer.ist.psu.edu/sudan97decoding.html`.
44 M. Sudan. List decoding: Algorithms and applications. *SIGACT News*, 31(1):16–27, 2000.
45 M. Sudan, L. Trevisan, and S. P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.
46 Terence Tao. *Higher Order Fourier Analysis*, volume 142 of *Graduate Studies in Mathematics*. American Mathematical Society, 2012.
47 Terence Tao and Tamar Ziegler. The inverse conjecture for the Gowers norm over finite fields via the correspondence principle. *Analysis & PDE*, 3(1):1–20, 2010.
48 Terence Tao and Tamar Ziegler. The inverse conjecture for the Gowers norm over finite fields in low characteristic. *Ann. Comb.*, 16(1):121–188, 2012.
49 Andre Weil. Sur les courbes algébriques et les varietes qui s'en deduisent. *Actualites Sci. et Ind.*, 1041, 1948.
50 J. Wozencraft. List decoding. Technical Report 48:90-95, Quarterly Progress Report, Research Laboratory of Electronics, MIT, 1958.
51 Yuichi Yoshida. A characterization of locally testable affine-invariant properties via decomposition theorems. In *Proc. 46th Annual ACM Symposium on the Theory of Computing*, pages 154–163, 2014.

Bounded Independence vs. Moduli[*]

Ravi Boppana[1], Johan Håstad[†2], Chin Ho Lee[‡3], and Emanuele Viola[§4]

1 **Department of Mathematics, Massachusetts Institute of Technology, Cambridge, USA**
2 **KTH-Royal Institute of Technology, Stockholm, Sweden**
3 **College of Computer and Information Science, Northeastern University, Boston, USA**
4 **College of Computer and Information Science, Northeastern University, Boston, USA**

Abstract

Let $k = k(n)$ be the largest integer such that there exists a k-wise uniform distribution over $\{0,1\}^n$ that is supported on the set $S_m := \{x \in \{0,1\}^n : \sum_i x_i \equiv 0 \bmod m\}$, where m is any integer. We show that $\Omega(n/m^2 \log m) \leq k \leq 2n/m + 2$. For $k = O(n/m)$ we also show that any k-wise uniform distribution puts probability mass at most $1/m + 1/100$ over S_m. For any fixed odd m there is $k \geq (1 - \Omega(1))n$ such that any k-wise uniform distribution lands in S_m with probability exponentially close to $|S_m|/2^n$; and this result is false for any even m.

1998 ACM Subject Classification F.0 Theory of Computation

Keywords and phrases Bounded independence, Modulus

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.24

1 Introduction and our results

A distribution on $\{0,1\}^n$ is k-wise uniform if any k bits are uniform in $\{0,1\}^k$. Researchers have analyzed various classes of tests that cannot distinguish distributions with k-wise uniformity from uniform. Such tests include (combinatorial) rectangles [8] (cf. [4]), bounded-depth circuits [1, 12, 2, 13], and halfspaces [6, 9, 7], to name a few. We say that such tests are *fooled* by distributions with bounded independence.

In this work we consider the mod m tests, defined next.

▶ **Definition 1.** For an input length n, and an integer m, we define the set $S_m := \{x \in \{0,1\}^n : \sum_i x_i \equiv 0 \bmod m\}$.

These tests have been intensely studied at least since circuit complexity theory hit the wall of gates computing mod m for composite m in the 80's. However, the effect of bounded independence on mod m tests does not seem to have been known before this paper.

Our first main result is that there exist distributions with linear uniformity that are supported on S_m.

[*] This work is done in part while CHL and EV were visiting Harvard University, with support from Salil Vadhan's Simons Investigator grant, and in part while JH, CHL and EV were at the Simons Institute for the Theory of Computing.
[†] Johan Håstad is supported by the Swedish Research Council.
[‡] Chin Ho Lee is supported by NSF grant CCF-1319206.
[§] Emanuele Viola is supported by NSF grant CCF-1319206.

▶ **Theorem 2.** *There exists a $c > 0$ such that the following holds.*

For every integer $m \geq 2$, there exists a $k \geq cn/m^2 \log m$ and a k-wise uniform distribution over $\{0,1\}^n$ that is supported on S_m.

This proves a conjecture in [10] where this question is also raised. Their motivation was a study of the "mod 3" dimension of k-wise uniform distributions, started in [11], which is the dimension of the space spanned by the support of the distribution over GF(3). [10] shows that $k = 100 \log n$-wise uniformity with dimension $\leq n^{0.49}$ would have applications to pseudorandomness. It also exhibits a distribution with dimension $n^{0.72}$ and uniformity $k = 2$. Theorem 2 yields a distribution with dimension $n - 1$ and $\Omega(n)$-wise uniformity.

We then prove three results, summarized in the next theorem, that show that k-wise uniformity does fool mod m when k is large. (1) shows that the largest possible value of k in Theorem 2 is $k \leq 2(n+1)/m + 2 \leq (1 - \Omega(1))n$. (2) shows that when k is larger than $(1-\gamma)n$ for a constant γ depending only on m then k-wise uniformity fools S_m with exponentially small error when m is odd. The proof of (2) however does not carry to the setting of $k < n/2$, for any m. So we establish (3) which gives a worse error bound but allows for k to become smaller for larger m, specifically $k = O(n/m)$ for constant error. The error bound in (3) and the density of S_m are such that (3) only provides a meaningful upper bound on the probability that the k-wise uniform distribution lands in S_m, but not a lower bound. In fact, we conjecture that no lower bound is possible in the sense that there is $c > 0$ such that for every m there is a cn-wise uniform distribution supported on the complement of S_m.

The combination of (2) and (3) implies that for $k = \min\{O(n/m), (1 - \Omega(1))n\}$ any k-wise uniform distribution puts probability mass at most $1/m + 1/100$ over S_m for odd m.

▶ **Theorem 3.** *Let m be an integer.*

(1) *For $k \geq 2n/m + 2$, a k-wise uniform distribution over $\{0,1\}^n$ cannot be supported on S_m.*

(2) *Suppose m is odd, then there is a $\gamma > 0$ depending only on m such that for any $(1-\gamma)n$-wise uniform distribution D over $\{0,1\}^n$, $|\Pr[D \in S_m] - |S_m|/2^n| \leq 2^{-\gamma n}$.*

(3) *There exists a universal constant c such that for every $\varepsilon > 0$, $n \geq cm^2 \log(m/\varepsilon)$, and any $c(n/m)(1/\varepsilon)^2$-wise uniform distribution D over $\{0,1\}^n$, $\Pr[D \in S_m] \leq |S_m|/2^n + \varepsilon$.*

In our results the sum s of n bits $x_i \in \{0,1\}$ is constrained to be divisible by m. This setting was chosen for convenience, but our techniques apply in greater generality. For example we obtain the same results if we instead constrain s to be c mod m for any fixed c.

We also note that (2) is false for any even m because the uniform distribution on S_2 has uniformity $k = n - 1$ but puts about $2/m$ mass on S_m, a set which as we shall see later (cf. Remark 7) has density about $1/m$.

Organization

Theorem 3 is a little easier to prove than Theorem 2, but uses overlapping lemmas. So we start by proving Theorem 3 in Section 2. Then in Section 3 we prove Theorem 2.

2 Proof of Theorem 3

In this section we prove Theorem 3. We start with the following theorem which will give (1) in Theorem 3 as a corollary.

▶ **Theorem 4.** *Let $I \subseteq \{0,1,\ldots,n\}$ be a subset of size $|I| \le n/2$. There does not exist a $2|I|$-wise uniform distribution on $\{0,1\}^n$ that is supported on $S := \{x \in \{0,1\}^n : \sum_i x_i \in I\}$.*

Proof. Suppose there exists such a distribution D. Consider the n-variate nonzero real polynomial p defined by

$$p(x) := \prod_{i\in I}(-i + \sum_{j=1}^{n} x_j).$$

Note that $p(x) = 0$ when $x \in S$. And so $\mathrm{E}[p^2(D)] = 0$ in particular. However, since p^2 has degree at most $2|I|$, we have $\mathrm{E}[p^2(D)] = \mathrm{E}[p^2(U)] > 0$, where U is the uniform distribution over $\{0,1\}^n$, a contradiction. ◀

Proof of (1) in Theorem 3. When I corresponds to the mod m test S_m, $|I| \le n/m+1$. ◀

We now move to (2) in Theorem 3. First we prove a lemma that estimates the sum $\sum_{x\in S_m}(-1)^{\sum_{i=1}^k x_i}$. Similar bounds have been established elsewhere, cf. e.g. Theorem 2.9 in [15], but we do not know of a reference with an explicit dependence on m, which will be used in the next section. (2) follows from bounding above the tail of the Fourier coefficients of the indicator function of S_m.

▶ **Lemma 5.** *For any $1 \le k \le n-1$, $|\sum_{x\in S_m}(-1)^{\sum_{i=1}^k x_i}| \le 2^n \left(\cos\frac{\pi}{2m}\right)^n$, while for $k = 0$ $|\sum_{x\in S_m}(-1)^{\sum_{i=1}^k x_i} - 2^n/m| \le 2^n \left(\cos\frac{\pi}{2m}\right)^n$. For odd m the first bound also holds for $k = n$.*

Proof. Consider an expansion of

$$p(y) = (1-y)^k(1+y)^{n-k}$$

into 2^n terms indexed by $x \in \{0,1\}^n$ where $x_i = 0$ indicates that we take the term 1 from the i'th factor. It is easy to see that the coefficient of y^d is $\sum_{|x|=d}(-1)^{\sum_{i=1}^k x_i}$. Denote $\zeta := e^{2\pi i/m}$ as the m-th root of unity. Recall the identity

$$\frac{1}{m}\sum_{j=0}^{m-1}\zeta^{jd} = \begin{cases} 1 & \text{if } d \equiv 0 \bmod m \\ 0 & \text{otherwise.}\end{cases}$$

Thus the sum we want to bound is equal to

$$\frac{1}{m}\sum_{j=0}^{m-1} p(\zeta^j).$$

Note that $p(\zeta^0) = p(1) = 0$ for $k \neq 0$ while for $k = 0$, $p(\zeta^0) = 2^n$. For the other terms we have the following bound.

▶ **Claim 6.** *For $1 \le j \le m-1$, $|p(\zeta^j)| \le 2^n \left(\cos\frac{\pi}{2m}\right)^k \left(\cos\frac{\pi}{m}\right)^{n-k}$.*

Proof. As $|1 + e^{i\theta}| = 2|\cos(\theta/2)|$ and $|1 - e^{i\theta}| = 2|\sin(\theta/2)|$ we have

$$\begin{aligned} |p(\zeta^j)| &= |1-\zeta^j|^k|1+\zeta^j|^{n-k} \\ &= 2^n\left(\sin\frac{j\pi}{m}\right)^k\left(\cos\frac{j\pi}{m}\right)^{n-k} \\ &\le 2^n\left(\cos\frac{\pi}{2m}\right)^k\left(\cos\frac{\pi}{m}\right)^{n-k}, \end{aligned}$$

where the last inequality holds for odd m because (1) $\sin\frac{j\pi}{m}$ is largest when $j = \frac{m-1}{2}$ or $j = \frac{m+1}{2}$, (2) $\sin(\frac{\pi}{2} - x) = \cos x$, and (3) $\cos\frac{j\pi}{m}$ is largest when $j = 1$ or $j = m - 1$. For even m the term with $j = m/2$ is 0, as in this case we are assuming that $k < n$, and the bounds for odd m are valid for the other terms. ◀

Therefore, for $k \neq 0$ we have

$$\left|\sum_{x\in S_m} (-1)^{\sum_{i=1}^k x_i}\right| = \frac{m-1}{m} \cdot 2^n \left(\cos\frac{\pi}{2m}\right)^k \left(\cos\frac{\pi}{m}\right)^{n-k} \leq 2^n \left(\cos\frac{\pi}{2m}\right)^k \left(\cos\frac{\pi}{m}\right)^{n-k},$$

and we complete the proof using the fact that $\cos(\pi/m) \leq \cos(\pi/2m)$. For $k = 0$ we also need to include the term $p(1) = 2^n$ which divided by m gives the term $2^n/m$. ◀

▶ Remark 7. Clearly the lemma for $k = 0$ simply is the well known fact that the cardinality of S_m is very close to $2^n/m$. Equivalently, if x is uniform in $\{0,1\}^n$ then the probability that $\sum_i x_i \in S_m$ is very close to $1/m$. The same holds for the probability that $\sum_i x_i \equiv c \bmod m$ for any fixed c. This can be seen by using the polynomial $y^{-c}p(y)$ in the above proof.

Proof of (2) in Theorem 3. Let $f : \{0,1\}^n \to \{0,1\}$ be the characteristic function of S_m. We first bound above the nonzero Fourier coefficients of f. Let $S = S_m$. By Lemma 5, we have for any β with $|\beta| = k > 0$,

$$|\hat{f}_\beta| = 2^{-n} \sum_{x\in S} (-1)^{\sum_{i=1}^k x_i} \leq \left(\cos\frac{\pi}{2m}\right)^n \leq 2^{-\alpha n},$$

where $\alpha = -\ln\cos(\pi/2m)$ depends only on m. Thus, if D is k-wise uniform,

$$\begin{aligned}|\mathrm{E}[f(D)] - \mathrm{E}[f(U)]| &\leq \sum_{|\beta|>k} |\hat{f}_\beta| \cdot |\mathrm{E}_{x\sim D}[(-1)^{\sum x_i\beta_i}]| \\ &\leq \sum_{|\beta|>k} |\hat{f}_\beta| \\ &\leq 2^{-\alpha n} \sum_{t=k+1}^{n} \binom{n}{t} \\ &= 2^{-\alpha n} \sum_{t=0}^{n-k-1} \binom{n}{t}.\end{aligned}$$

For $k \geq (1-\delta)n$, we have an upper bound of $2^{n(H(\delta)-\alpha)}$. Pick δ small enough so that $H(\delta) \leq \alpha/2$. The result follows by setting $\gamma := \min\{\alpha/2, \delta\}$. ◀

Note that the above proof fails when m is even as we cannot handle the term with $|\beta| = n$. Finally, we prove (3) in Theorem 3. We use approximation theory.

Proof of (3) in Theorem 3. Let $f\colon \{0,1\}^n \to \{0,1\}$ be the characteristic function of S_m. The proof amounts to exhibiting a real polynomial p in n variables of degree $d = c(n/m)(1/\varepsilon)^2$ such that $f(x) \leq p(x)$ for every $x \in \{0,1\}^n$, and $\mathrm{E}[p(U)] \leq \varepsilon$ for U uniform over $\{0,1\}^n$. To see that this suffices, note that $\mathrm{E}[p(U)] = \mathrm{E}[p(D)]$ for any distribution D that is d-wise uniform. Using this and the fact that f is non-negative, we can write

$$0 \leq \mathrm{E}[f(U)] \leq \mathrm{E}[p(U)] \leq \varepsilon \quad \text{and} \quad 0 \leq \mathrm{E}[f(D)] \leq \mathrm{E}[p(D)] \leq \varepsilon.$$

Hence, $|\mathrm{E}[f(U)] - \mathrm{E}[f(D)]| \leq \varepsilon$. This is the method of sandwiching polynomials from [1].

Let us write $f = g(\sum_i x_i/n)$, for $g\colon \{0, 1/n, \ldots, 1\} \to \{0, 1\}$. We exhibit a univariate polynomial q of degree d such that $g(x) \leq q(x)$ for every x, and the expectation of q under the binomial distribution is at most ε. The polynomial p is then $q(\sum_i x_i/n)$.

Consider the continuous, piecewise linear function $s : [-1, 1] \to [0, 1]$ defined as follows. The function is always 0, except at intervals of radius a/n around the inputs x where g equals 1, i.e., inputs x such that nx is divisible by m. In those intervals it goes up and down like a 'Λ', reaching the value of 1 at x. We set $a = \varepsilon m/10$.

By Jackson's theorem, see e.g. [3, Theorem 7.4] or [5], for a degree $d = O(n\varepsilon^{-1}a^{-1}) = O(n\varepsilon^{-2}m^{-1})$, there exists a univariate polynomial q' of degree d that approximates s with pointwise error $\varepsilon/10$. Our polynomial q is defined as $q := q' + \varepsilon/10$.

It is clear that $g(x) \leq q(x)$ for every $x \in \{0, 1/n, \ldots, 1\}$. It remains to estimate $\mathrm{E}[q(U)]$.

As q' is a good approximation of s we have $\mathrm{E}[q(U)] \leq 2\varepsilon/10 + \mathrm{E}[s(U)]$. We noted in Remark 7 that the remainder modulo m of $\sum x_i$ is δ-close to uniform for $\delta = \cos(\pi/2m)^n = e^{-O(n/m^2)}$. Now the function s, as a function of $\sum x_i$, is a periodic function with period m and if we feed the uniform distribution over $\{0, 1/n, \ldots, m/n\}$ into s we have $\mathrm{E}[s] \leq \varepsilon/10$. It follows that if n is at least a large constant times $m^2(\log(1/\varepsilon) + \log m)$, we have $\mathrm{E}[s(U)] \leq 2\varepsilon/10$ and we conclude that $\mathrm{E}[q(U)] \leq 4\varepsilon/10$. ◀

3 Proof of Theorem 2

In this section we prove Theorem 2. Let I be a subset of $\{0, 1, \ldots, n-1, n\}$ and $S \subseteq \{0, 1\}^n$ be the subset of strings whose sum $\sum_i x_i$ belongs to I. Let U_S be the uniform distribution over S. We are going to construct a k-wise uniform distribution starting from U_S and changing the weights of $k+1$ slices of the Hamming cube. In particular, our distribution will be symmetric. We note that since S is symmetric, if there is a k-wise uniform distribution supported on it then by a simple symmetrization argument there must also be a symmetric one.

Let ε_t be the bias of a parity of size t under U_S, i.e., $\varepsilon_t := E_{x \in U_S}[(-1)^{\sum_{i=1}^t x_i}]$. Note that because we are working with symmetric distributions, all parities of the same size have the same bias. Now let $\varepsilon(t, \ell)$ be the bias of a parity of size t over the uniform distribution on strings that sum to ℓ. Note that $\varepsilon(t, \ell)$ is a scaled version of the Kravchuk polynomial of degree t in the variable ℓ.

We note that $\varepsilon_t = \sum_{\ell \in I} \Pr_{x \sim U_S}[\sum_j x_j = \ell] \cdot \varepsilon(t, \ell)$.

Now let $a_0 < a_1 < \cdots < a_k$ be $k+1$ points in I that are closest to $n/2$ and let i^* be an index that maximizes $|a_i - \frac{n}{2}|$. Finally let p_i be the probability over x drawn from U_S that x sums to a_i.

We are going to change the p_i to $p_i - \Delta_i$ with the goal of making ε_t zero for every $1 \leq t \leq k$. The effect of the substitution on ε_t is to decrease it by $\sum_{0 \leq i \leq k} \Delta_i \varepsilon(t, a_i)$.

Thus our goal is to find Δ_i's so that

$$\sum_{i=0}^{k} \Delta_i \varepsilon(t, a_i) = \varepsilon_t, \quad \forall t \in \{1, 2, \ldots, k\}$$

$$\sum_{i=0}^{k} \Delta_i = 0,$$

$$0 \leq p_i - \Delta_i \leq 1, \quad \forall i \in \{0, \ldots, k\}.$$

Let M be the $(k+1) \times (k+1)$ matrix $M_{t,i} := \varepsilon(t, a_i)$ where $t, i \in \{0, \ldots, k\}$. Let $\Delta :=$

$(\Delta_0, \ldots, \Delta_k)^T$ and $b := (0, \varepsilon_1, \ldots, \varepsilon_k)^T$. Then the first two conditions form the linear system

$$M\Delta = b.$$

We will show that there is a unique solution Δ to this system.

To satisfy the third condition, note that p_{i^*} is the smallest among all the p_i's. It will also be the case that $p_{i^*} \leq 1/2$. Thus if $\|\Delta\|_\infty \leq p_{i^*}$ we will also satisfy the third condition and have a k-wise uniform distribution supported on S.

Consider the expression $n^{-t}(\sum_{j=1}^{n}(-1)^{x_j})^t$. If we expand this, cancel factors that appear twice, and collect terms, we can rewrite it as

$$n^{-t}(\sum_{j=1}^{n}(-1)^{x_j})^t = \sum_{r=0}^{t} \gamma_{t,r} \binom{n}{r}^{-1} \sum_{|\beta|=r} (-1)^{\sum x_i \beta_i},$$

for some choice of non-negative values $\gamma_{t,r}$, which by plugging in $x_1 = x_2 = \ldots = x_n = 0$ can be seen to satisfy $\sum_{r=0}^{t} \gamma_{t,r} = 1$.

Let $\alpha_i := (n - 2a_i)/n$. Taking expectation in the above equation over all the x's with sum equal to a_i we have for every $i \in \{0, 1, \ldots, k\}$,

$$\alpha_i^t = ((n-2a_i)/n)^t = \sum_{r=0}^{t} \gamma_{t,r} \binom{n}{r}^{-1} \sum_{|\beta|=r} \mathrm{E}[(-1)^{\sum x_i \beta_i}] = \sum_{r=0}^{t} \gamma_{t,r} \varepsilon(r, a_i). \tag{A}$$

Let M_r be the r-th row of M. We construct a new matrix V from M by applying the following row operations R to M: For every t, set $V_t = \sum_{r=0}^{t} \gamma_{t,r} M_r$. It follows from equation (A) that $V_{t,i} = \alpha_i^t$, and so $V = RM$ is a Vandermonde matrix, which is invertible. Hence,

$$\Delta = V^{-1} R b$$

is a unique solution.

Therefore it suffices to show that $\|\Delta\|_\infty \leq p_{i^*}$. Note that $\|\Delta\|_\infty \leq \|V^{-1}\|_\infty \|Rb\|_\infty$, where the ∞ norm of a matrix is the maximum sum of the absolute values along any one row.

Moreover, since $(Rb)_t = \sum_{r=0}^{t} \gamma_{t,r} b_r$ and $\sum_{r=0}^{t} \gamma_{t,r} = 1$, we have $\|Rb\|_\infty \leq \|b\|_\infty$. Hence, it suffices to bound above $\|V^{-1}\|_\infty$ and $\|b\|_\infty$.

Roadmap for the following claims

To get an idea of the following claims, consider the case $m = 3$ and $k = o(n)$. We first show in Claim 8 that $\|V^{-1}\|_\infty \leq 2^{o(n)}$. Then we find it convenient to bound $\|b\|_\infty$ and p_{i^*} multiplied by $|S|$. We show that $|S| p_{i^*} \geq 2^{n(1-o(1))}$ in Claim 9. We note that Claims 8, 9 and 10 hold for any symmetric subset S. Finally, in Claim 11 we use the definition of S to obtain bounds on a_{i^*} and b, and show that $|S| \|b\|_\infty \leq (2 - \Omega(1))^n$. Altogether,

$$\|V^{-1}\|_\infty |S| \|b\|_\infty \leq 2^{o(n)} (2 - \Omega(1))^n \leq 2^{n(1-\Omega(1))} \leq |S| p_{i^*},$$

as desired.

▶ **Claim 8.** $\|V^{-1}\|_\infty \leq (k+1)(\frac{4en}{k})^k$.

Proof. Since V is a Vandermonde matrix, we can specify the entries of its inverse explicitly. As shown in e.g. [14] we have

$$V^{-1}_{i,k-j} = (-1)^{k-j} \left(\sum_{\substack{|\beta|=j \\ i \notin \beta}} \alpha^\beta \right) \cdot \left(\prod_{s \neq i} (\alpha_s - \alpha_i)^{-1} \right).$$

We now give an upper bound on each of the factors on the R.H.S.

Bounding $\sum_{|\beta|=j, i\notin\beta} \alpha^\beta$

Since $|\alpha_i| \le 1$, this is bounded by the number of terms, $\binom{k}{j}$, and hence by 2^k.

Bounding $\prod_{s\neq i}(\alpha_s - \alpha_i)^{-1}$

Since the difference between every pair of distinct a_i, a_j is at least 1, we have

$$\prod_{s\neq i}(a_s - a_i) \ge (k/2)!^2$$

when k is even and is at least $(\frac{k+1}{2})(\frac{k-1}{2})!^2$ when k is odd. By a crude form of Stirling's formula, $n! \ge (n/e)^n$, and so we get the lower bound $(k/2e)^k$ in either case. Hence,

$$\prod_{s\neq i}(\alpha_s - \alpha_i)^{-1} \le n^k \prod_{s\neq i}(a_s - a_i)^{-1} \le (\frac{2en}{k})^k.$$

Putting the bounds together, we have

$$\|V^{-1}\|_\infty \le (k+1)\max_{i,j}|V^{-1}_{i,j}| \le (k+1)(\frac{4en}{k})^k.$$ ◀

Now we give a lower bound on p_{i^*}.

▶ **Claim 9.** $p_{i^*}|S| \ge \frac{2^{n\left(1-\alpha_{i^*}^2\right)}}{n+1}$.

Proof. Using the inequalities $\binom{n}{i} \ge \frac{2^{nH(i/n)}}{n+1}$ and $H(\frac{1-\varepsilon}{2}) \ge 1-\varepsilon^2$, we have

$$p_{i^*}|S| = \binom{n}{a_{i^*}} \ge \frac{2^{nH\left(\frac{1-\alpha_{i^*}}{2}\right)}}{n+1} \ge \frac{2^{n\left(1-\alpha_{i^*}^2\right)}}{n+1}.$$ ◀

Therefore,

$$\frac{p_{i^*}|S|}{\|V^{-1}\|_\infty} \ge \frac{2^{n\left(1-\alpha_{i^*}^2\right)}}{(n+1)(k+1)(\frac{4en}{k})^k} \ge e^{nf(k,n,a_{i^*})},$$

where

$$f(k,n,a_{i^*}) := \ln 2 \cdot \left(1-\alpha_{i^*}^2\right) - \frac{k}{n}\left(\ln\frac{4en}{k}\right) - o(1).$$

We conclude with the following claim.

▶ **Claim 10.** *If* $e^{nf(k,n,a_{i^*})} \ge \max_{1\le t\le k}\sum_{x\in S}(-1)^{\sum_{i=1}^t x_i}$*, then there exists a* k*-wise uniform distribution supported on* S*.*

Proof. We just showed

$$\frac{p_{i^*}|S|}{\|V^{-1}\|_\infty} \ge e^{nf(k,n,a_{i^*})} \ge \max_{1\le t\le k}\sum_{x\in S}(-1)^{\sum_{i=1}^t x_i} = \|b\|_\infty|S|.$$

Hence, $\|\Delta\|_\infty \le \|V^{-1}\|_\infty\|b\|_\infty \le p_{i^*}$. ◀

3.1 Zero modulo m

We have that S_m consists of all strings with $\sum x_i \equiv 0 \bmod m$. If follows that $|\alpha_{i^*}| \leq (k+1)m/2n$. We now give an upper bound on $\|b\|_\infty |S|$.

▶ **Claim 11.** *$\|b\|_\infty |S| \leq e^{ng(n,m)}$, where $g(n,m) := \ln 2 - \frac{1}{2}\left(\frac{\pi}{2m}\right)^2$.*

Proof. Note that $\|b\|_\infty |S| = \sum_{x\in S}(-1)^{\sum_{i=1}^k x_i}$. By Lemma 5,

$$\sum_{x\in S}(-1)^{\sum_{i=1}^k x_i} \leq 2^n\left(\cos\frac{\pi}{2m}\right)^n \leq e^{ng(n,m)},$$

where in the last two inequalities we used the fact that $\ln\cos(x) \leq -\frac{x^2}{2}$ for $x \in [0, \pi/2)$. ◀

We are now ready to prove Theorem 2.

Proof of Theorem 2. Recall that $|\alpha_{i^*}| \leq (k+1)m/2n$. By Claim 11 and Claim 10, it suffices to show that $f(k, n, a_{i^*}) - g(n,m)$ is positive, where recall

$$\begin{aligned} f(k,n,a_i^*) &= \ln 2\cdot\left(1-\alpha_{i^*}^2\right) - \frac{k}{n}\left(\ln\frac{4en}{k}\right) - o(1) \\ &\geq \ln 2\cdot\left(1-(\frac{(k+1)m}{2n})^2\right) - \frac{k}{n}\left(\ln\frac{4en}{k}\right) - o(1) \end{aligned}$$

and

$$g(n,m) := \ln 2 - \frac{1}{2}\left(\frac{\pi}{2m}\right)^2.$$

Indeed, we have

$$f(k,n,a_{i^*}) - g(n,m) \geq \frac{1}{2}\left(\frac{\pi}{2m}\right)^2 - \frac{k}{n}\left(\ln\frac{4en}{k}\right) - \ln 2\cdot\left(\frac{(k+1)m}{2n}\right)^2 - o(1),$$

and choosing $k = \frac{\varepsilon n}{m^2 \ln m}$ for a sufficiently small ε makes this quantity positive. ◀

References

1 Louay M. J. Bazzi. Polylogarithmic independence can fool DNF formulas. *SIAM J. Comput.*, 38(6):2220–2272, 2009.

2 Mark Braverman. Polylogarithmic independence fools AC^0 circuits. *J. of the ACM*, 57(5), 2010.

3 Neal Carothers. A short course on approximation theory. Available at http://personal.bgsu.edu/~carother/Approx.html.

4 Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Improved algorithms via approximations of probability distributions. *J. Comput. System Sci.*, 61(1):81–107, 2000. doi:10.1006/jcss.1999.1695.

5 E. Cheney. *Introduction to approximation theory.* McGraw-Hill, New York, New York, 1966.

6 Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco A. Servedio, and Emanuele Viola. Bounded independence fools halfspaces. *SIAM J. on Computing*, 39(8):3441–3462, 2010.

7 Ilias Diakonikolas, Daniel Kane, and Jelani Nelson. Bounded independence fools degree-2 threshold functions. In *51th IEEE Symp. on Foundations of Computer Science (FOCS)*. IEEE, 2010.

8 Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Velickovic. Efficient approximation of product distributions. *Random Struct. Algorithms*, 13(1):1–16, 1998.

9 Parikshit Gopalan, Ryan O'Donnell, Yi Wu, and David Zuckerman. Fooling functions of halfspaces under product distributions. In *25th IEEE Conf. on Computational Complexity (CCC)*, pages 223–234. IEEE, 2010.

10 Chin Ho Lee and Emanuele Viola. Some limitations of the sum of small-bias distributions. Available at `http://www.ccs.neu.edu/home/viola/`, 2015.

11 Raghu Meka and David Zuckerman. Small-bias spaces for group products. In *13th Workshop on Randomization and Computation (RANDOM)*, volume 5687 of *Lecture Notes in Computer Science*, pages 658–672. Springer, 2009.

12 Alexander A. Razborov. A simple proof of Bazzi's theorem. *ACM Transactions on Computation Theory (TOCT)*, 1(1), 2009.

13 Avishay Tal. Tight bounds on The Fourier Spectrum of AC^0. *Electronic Colloquium on Computational Complexity*, Technical Report TR14-174, 2014.

14 L. Richard Turner. Inverse of the Vandermonde matrix with applications, 1966. NASA technical note D-3547 available at `http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19660023042.pdf`.

15 Emanuele Viola and Avi Wigderson. Norms, XOR lemmas, and lower bounds for polynomials and protocols. *Theory of Computing*, 4:137–168, 2008.

Approximating Subadditive Hadamard Functions on Implicit Matrices

Vladimir Braverman*[1], Alan Roytman†[2], and Gregory Vorsanger‡[3]

1 **Department of Computer Science, Johns Hopkins University, Baltimore, USA**
`vova@cs.jhu.edu`

2 **School of Computer Science, Tel-Aviv University, Israel**
`alan.roytman@cs.tau.ac.il`

3 **Department of Computer Science, Johns Hopkins University, Baltimore, USA**
`gregvorsanger@jhu.edu`

Abstract

An important challenge in the streaming model is to maintain small-space approximations of entrywise functions performed on a matrix that is generated by the outer product of two vectors given as a stream. In other works, streams typically define matrices in a standard way via a sequence of updates, as in the work of Woodruff [22] and others. We describe the matrix formed by the outer product, and other matrices that do not fall into this category, as implicit matrices. As such, we consider the general problem of computing over such implicit matrices with Hadamard functions, which are functions applied entrywise on a matrix. In this paper, we apply this generalization to provide new techniques for identifying independence between two data streams. The previous state of the art algorithm of Braverman and Ostrovsky [9] gave a $(1 \pm \epsilon)$-approximation for the L_1 distance between the joint and product of the marginal distributions, using space $O(\log^{1024}(nm)\epsilon^{-1024})$, where m is the length of the stream and n denotes the size of the universe from which stream elements are drawn. Our general techniques include the L_1 distance as a special case, and we give an improved space bound of $O(\log^{12}(n)\log^2(\frac{nm}{\epsilon})\epsilon^{-7})$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Streaming Algorithms, Measuring Independence, Hadamard Functions, Implicit Matrices

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.25

1 Introduction

Measuring independence is a fundamental statistical problem that is well studied in computer science. Traditional non-parametric methods of testing independence over empirical data usually require space complexity that is polynomial in either the support size or input size. With large datasets, these space requirements may be impractical, and designing small-space algorithms becomes desirable.

Measuring independence is a classic problem in the field of statistics (see Lehmann [17]) as well as an important problem in databases. Further, the process of reading in a two-column

* This work was sponsored in part by the National Science Foundation under Grant No. 1447639 and the Google Faculty Award.

† The work of Alan Roytman was partially supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement number 337122, and in part by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11).

‡ This work was supported in part by Raytheon BBN Technologies.

database table can be viewed as a stream of pairs. Thus, the streaming model is a natural choice when approximating pairwise independence as memory is limited. Indeed, identifying correlations between database columns by measuring the level of independence between columns is of importance to the database and data warehouse community (see, e.g., [19] and [16], respectively).

In this paper we provide new techniques for measuring independence between two data streams and present new tools to expand existing techniques. The topic of independence was first studied in the streaming model by Indyk and McGregor [15] where the authors gave an optimal algorithm for approximating the L_2 distance between the joint and product of the marginal distributions of two random variables which generate a stream. In their work, they provided a sketch that is pairwise independent, but not 4-wise independent, so analysis similar to that of Alon, Matias, and Szegedy [3] cannot be applied directly. This work was continued by Braverman and Ostrovsky [9], where the authors considered comparing among a stream of k-tuples and provided the first $(1 \pm \epsilon)$-approximation for the L_1 distance between the joint and product of the marginal distributions. Their algorithm is currently the best known space bound, and uses $O(\frac{1}{\epsilon^{1024}} \log^{1024}(nm))$ space for $k = 2$, where m is the length of the stream and n denotes the size of the universe from which stream elements are drawn. We present new methods, in the form of a general tool, that enable us to improve this bound to $O(\frac{1}{\epsilon^7} \log^{12}(n) \log^2(\frac{nm}{\epsilon}))$. In previous works, a central challenge has been maintaining an approximation of the matrix that is generated by the outer product of the two streaming vectors. As such, we consider computing functions on such an implicit matrix. While matrices have been studied previously in the streaming model (e.g., [22]), note that we cannot use standard linear sketching techniques, as the entries of the matrix are given implicitly and thus these methods do not apply directly.

Generalizing this specific motivating example, we consider the problem of obtaining a $(1 \pm \epsilon)$-approximation of the L_1 norm of the matrix $g[A]$, where $g[A]$ is the matrix A with a function g applied to it entrywise. Such mappings g are called *Hadamard* functions (see [12, 13]). Note that we sometimes abuse notation and apply the function g to scalar values instead of matrices (e.g., $g(a_{ij})$ where a_{ij} is the $(i,j)^{th}$ entry in matrix A). We require the scalar form of the function g to be even, subadditive, non-negative, and zero at the origin. We show that, given a blackbox $r(n)$-approximation of $\|g[A]\|_1 = \sum_i \sum_j g(a_{ij})$ (where a_{ij} is the $(i,j)^{th}$ entry in matrix A) and a blackbox $(1 \pm \epsilon)$-approximation of the aggregate of g applied entrywise to a vector obtained by summing over all rows, we are able to improve the $r(n)$-approximation to a $(1 \pm \epsilon)$-approximation (where $r(n)$ is a sufficiently large monotonically increasing function of n). Hence, we give a reduction for any such function g. Our reduction can be applied as long as such blackbox algorithms exist.

An interesting special case of our result is when the matrix is defined by the L_1 distance between the joint and product of the marginal distributions, which corresponds to measuring independence in data streams. Since such blackbox algorithms are known for L_1, not only does our framework generalize the problem of measuring independence according to the L_1 distance, but our algorithmic techniques also yield improved space bounds over the previous state of the art result [9]. Moreover, our framework would immediately translate improved space bounds for the blackbox algorithms to improved space bounds for the application of measuring independence. Note that, for L_p where $0 < p < 1$, such blackbox algorithms are not known. If such algorithms for the L_p distance were to be designed, our reductions work and can be applied. While there are a variety of ways to compute distances between distributions, the L_p distance is of particular significance as evidenced in [14].

Motivating Problem

We begin by presenting our motivating problem, which concerns (approximately) measuring the distance between the joint and product of the marginal distributions of two random variables. That is, we attempt to quantify how close two random variables X and Y over a universe $[n] = \{1, \ldots, n\}$ are to being independent. There are many ways to measure the distance between distributions, but we focus on the L_1 distance. Recall that two random variables X and Y are independent if we have $\Pr[X = i \wedge Y = j] = \Pr[X = i] \Pr[Y = j]$ for every i and j. In our model, we have a data stream D which is presented as a sequence of m pairs $d_1 = (i_1, j_1), d_2 = (i_2, j_2), \ldots, d_m = (i_m, j_m)$. Each pair $d_k = (i_k, j_k)$ consists of two integers taken from the universe $[n]$.

Intuitively, we imagine that the two random variables X and Y over the universe $[n]$ generate these pairs, and in particular, the frequencies of each pair (i, j) define an empirical joint distribution, which is the fraction of pairs that equal (i, j). At the same time, the stream also defines the empirical marginal distributions $\Pr[X = i], \Pr[Y = j]$, namely the fraction of pairs of the form $(i, \cdot)$ and $(\cdot, j)$, respectively. We note that, even if the pairs are actually generated from two independent sources, it may not be the case that the empirical distributions reflect this fact, although for sufficiently long streams the joint distribution should approach the product of the marginal distributions for each i and j. This fundamental problem has received considerable attention within the streaming community, including the works of [15, 9]. We note that the main theoretical contribution of this paper is focused on a generalization of this problem. Nevertheless, this application serves as a very important motivation for our framework, and we explain how to apply our framework to it in Section 5. For the main problem we solve, please see Problem 2.

▶ **Problem 1.** *Let X and Y be two random variables which generate a stream of m pairs $d_1 = (i_1, j_1), \ldots, d_m = (i_m, j_m)$, where each $i_k, j_k \in [n]$ for all k. Define the frequencies $p_i = |\{k : d_k = (i, \cdot)\}|$ and $q_j = |\{k : d_k = (\cdot, j)\}|$ (i.e., the frequency with which i appears in the first coordinate and j appears in the second coordinate, respectively). Moreover, let $f_{ij} = |\{k : d_k = (i, j)\}|$ be the frequency with which the pair (i, j) appears in the stream. This naturally defines the joint distribution $\Pr[X = i \wedge Y = j] = \frac{f_{ij}}{m}$ and the product of the marginal distributions $\Pr[X = i] \Pr[Y = j] = \frac{p_i q_j}{m^2}$. The L_1 distance between the joint and product of the marginal distributions is given by:*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \left| \frac{f_{ij}}{m} - \frac{p_i q_j}{m^2} \right| .$$

If X and Y are independent, we should expect this sum to be close to 0, assuming the stream is sufficiently long. As a generalization to this problem, we can view the n^2 values which appear in the summation as being implicitly represented via an $n \times n$ matrix, where the $(i, j)^{th}$ entry is given by $\left| \frac{f_{ij}}{m} - \frac{p_i q_j}{m^2} \right|$. For the motivating problem, this matrix is given implicitly as it is not given up front and changes over time according to the data stream (each new pair in the stream may change multiple entries in the matrix). However, one can imagine settings in which these entries are defined through other means. In practice, we may still be interested in computing approximate statistics over such implicitly defined matrices.

Contributions and Techniques

Our main contributions in this paper make progress on two important problems:

Table 1 Comparing Approximation Ratios and Space Complexity.

Previous Work	L_1 approximation	Memory
IM08 [15]	$\log(n)$	$O\left(\frac{1}{\epsilon^2}\log\left(\frac{nm}{\epsilon}\right)\log\left(\frac{m}{\epsilon}\right)\right)$
BO10[1] [9]	$(1 \pm \epsilon)$	$O\left(\left(\frac{\log(nm)}{\epsilon}\right)^{1024}\right)$
Our Result	$(1 \pm \epsilon)$	$O\left(\frac{1}{\epsilon^7}\log^{12}(n)\log^2\left(\frac{nm}{\epsilon}\right)\right)$

1. For any subadditive even Hadamard function g where g is non-negative and $g(0) = 0$, given an implicitly defined $n \times n$ matrix A with entries a_{ij}, let $g[A]$ be the matrix where the $(i,j)^{th}$ entry is $g(a_{ij})$. We are the first to provide a general reduction framework for approximating $\|g[A]\|_1 = \sum_{i=1}^n \sum_{j=1}^n g(a_{ij})$ to within a $(1 \pm \epsilon)$-factor with constant success probability. More formally, suppose we have two blackbox algorithms with the following guarantees. One blackbox algorithm operates over the implicit matrix A and provides a very good ($\approx 1 \pm \epsilon$) approximation to $\|g[JA]\|_1 = \sum_{j=1}^n g(\sum_{i=1}^n a_{ij})$ except with inverse polylogarithmic probability, where $J = (1, \ldots, 1)$ is the row vector of dimension n with every entry equal to 1. The second blackbox algorithm operates over the implicit matrix A and solves the problem we wish to solve (i.e., approximating $\|g[A]\|_1$) with constant success probability, although it does so with a multiplicative approximation ratio of $r(n)$ (which may be worse than $(1 \pm \epsilon)$ in general). We show how to use these two blackbox algorithms to construct an algorithm that achieves a $(1 \pm \epsilon)$-approximation of $\|g[A]\|_1$. If S_1, S_2 denote the space used by the first and second blackbox algorithms, respectively, then our algorithm uses space $O\left(\frac{r^4(n)\log^8(n)}{\epsilon^5} \cdot (\log^2(n) + S_1 + \log(n) \cdot S_2)\right)$. We state this formally in Theorem 3.
2. Given the contribution above, it follows that setting $g(x) = |x|$ solves Problem 1, namely the problem of measuring how close two random variables are to being independent, as long as such blackbox algorithms exist. In particular, the work of Indyk [14] provides us with the first blackbox algorithm, and the work of [15] provides us with the second blackbox algorithm for this choice of g. Combining these results, we improve over the previous state of the art result of Braverman and Ostrovsky [9] and give improved bounds for measuring independence of random variables in the streaming model by reducing the space usage from $O\left((\frac{\log(nm)}{\epsilon})^{1024}\right)$ to $O\left(\frac{1}{\epsilon^7}\log^{12}(n)\log^2\left(\frac{nm}{\epsilon}\right)\right)$ (see Table 1).

Examples of such Hadamard functions which are subadditive, even, non-negative, and zero at the origin include $g(x) = |x|^p$, for any $0 < p \leq 1$. Note that our reduction in the first item can only be applied to solve the problem of approximating $\|g[A]\|_1$ if such blackbox algorithms exist, but for some functions g this may not be the case. As a direct example of the tools we present, we give a reduction for computing the L_p distance for $0 < p < 1$ between the joint and product of the marginal distributions in the streaming model (as this function is even and subadditive). However, to the best of our knowledge, such blackbox algorithms do not exist for computing the L_p distance. Thus, as a corollary to our main result, the construction of such space efficient blackbox algorithms would immediately yield a space efficient algorithm that measures independence according to the L_p distance.

[1] The paper of [9] provides a general bound for the L_1 distance for k-tuples, but we provide analysis for pairs of elements, $k = 2$, in this paper. The bound in the table is for $k = 2$.

Our techniques leverage concepts provided in [9, 15] and manipulates them to allow them to be combined with the Recursive Sketches data structure [11] to gain a large improvement compared to existing bounds. Note that we cannot use standard linear sketching techniques because the entries of the matrix are given implicitly. Moreover, the sketch of Indyk and McGregor [15] is pairwise independent, but not 4-wise independent. Therefore, we cannot apply the sketches of [3, 15] directly. We first present an algorithm, independent of the streaming model, for finding heavy rows of a matrix norm given an arbitrary even subadditive Hadamard function g. In order to do this, we first prove a key theorem regarding such Hadamard functions g which states that the quantity $\|g[JA]\|_1 = \sum_{j=1}^{n} g(\sum_{i=1}^{n} a_{ij})$ is a $(1 \pm \epsilon)$-approximation to the heavy row of the matrix $g[A]$ (if it exists). With this in mind, we show how to use the blackbox algorithm that yields an $r(n)$-approximation to $\|g[A]\|_1$ in order to identify when heavy rows exist in the matrix, and then use the other blackbox algorithm to obtain a $(1 \pm \epsilon)$-approximation of $\|g[JA]\|_1$ (which is in turn a $(1 \pm \epsilon)$-approximation to the heavy row, as just mentioned). These ideas form the foundation of our algorithm for approximating heavy rows. We then apply the Recursive Sum algorithm from [11] on top of our heavy rows algorithm to obtain our main result.

1.1 Related Work

In their seminal 1996 paper Alon, Matias, and Szegedy[3] provided an optimal space approximation for L_2. A key technical requirement of the sketch is the assumption of 4-wise independent random variables. This technique is the building block for measuring the independence of data streams using L_2 distances as well.

The problems of efficiently testing pairwise, or k-wise, independence were considered by Alon, Andoni, Kaufman, Matulef, Rubinfeld, and Xie [1]; Alon, Goldreich, and Mansour [2]; Batu, Fortnow, Fischer, Kumar, Rubinfeld, and White [4]; Batu, Kumar, and Rubinfeld [7]; Batu, Fortnow, Rubinfeld, Smith, and White [5, 6]. They addressed the problem of minimizing the number of samples needed to obtain a sufficient approximation, when the joint distribution is accessible through a sampling procedure.

In their 2008 work, Indyk and McGregor [15] provided exciting results for identifying the correlation of two streams, providing an optimal bound for determining the L_2 distance between the joint and product of the marginal distributions of two random variables.

In addition to the L_2 result, Indyk and McGregor presented a $\log(n)$-approximation for the L_1 distance. This bound was improved to a $(1 \pm \epsilon)$-approximation in the work of Braverman and Ostrovsky [9] in which they provided a bound of $O(\frac{1}{\epsilon^{1024}} \log^{1024}(nm))$ for pairs of elements. Further, they gave bounds for the comparison of multiple streaming vectors and determining k-wise relationships for L_1 distance. In addition, Braverman et al. [8] expanded the work of [15] to k dimensions for L_2. Recently, McGregor and Vu [18] studied a related problem regarding Bayesian networks in the streaming model.

Statistical distance, L_1, is one of the most fundamental metrics for measuring the similarity of two distributions. It has been the metric of choice in many of the above testing papers, as well as others such as Rubinfeld and Servedio [20]; Sahai and Vadhan [21]. As such, a main focus of this work is improving bounds for this measure in the streaming model.

2 Problem Definition and Notation

In this paper we focus on the problem of approximating even, subadditive, non-negative Hadamard functions which are zero at the origin on implicitly defined matrices (e.g., the streaming model implicitly defines matrices for us in the context of measuring independence). The main problem we study in this paper is the following:

▶ **Problem 2.** *Let g be any even, subadditive, non-negative Hadamard function such that $g(0) = 0$. Given any implicit matrix A, for any $\epsilon > 0$, $\delta > 0$, output a $(1 \pm \epsilon)$-approximation of $\|g[A]\|_1$ except with probability δ.*

We now provide our main theorem, which solves Problem 2.

▶ **Theorem 3.** *Let g be any even, subadditive, non-negative Hadamard function g where $g(0) = 0$, and fix $\epsilon > 0$. Moreover, let A be an arbitrary matrix, and J be the all 1's row vector $J = (1, \dots, 1)$ of dimension n. Suppose there are two blackbox algorithms with the following properties:*

1. *Blackbox Algorithm 1, for all $\epsilon' > 0$, returns a $(1 \pm \epsilon')$-approximation of $\|g[JA]\|_1$, except with probability δ_1.*
2. *Blackbox Algorithm 2 returns an $r(n)$-approximation of $\|g[A]\|_1$, except with probability δ_2 (where $r(n)$ is a sufficiently large monotonically increasing function of n).*

Then, there exists an algorithm that returns a $(1 \pm \epsilon)$-approximation of $\|g[A]\|_1$, except with constant probability. If Blackbox Algorithm 1 uses space $SPACE1(n, \delta_1, \epsilon')$, and Blackbox Algorithm 2 uses space $SPACE2(n, \delta_2)$, the resulting algorithm has space complexity

$$O\left(\frac{r^4(n)}{\epsilon^5}(\log^{10}(n) + \log^8(n)SPACE1(n, \delta_1, \epsilon') + \log^9(n)SPACE2(n, \delta_2))\right),$$

where $\epsilon' = \frac{\epsilon}{2}$, δ_1 is a small constant, and δ_2 is inverse polylogarithmic.

Note that we can reduce the constant failure probability to inverse polynomial failure probability via standard techniques, at the cost of increasing our space bound by a logarithmic factor. Observe that Problem 2 is a general case of Problem 1 where $g(x) = |x|$ (i.e., L_1 distance). In the streaming model, we receive matrix A implicitly, but we conceptualize the problem as if the matrix were given explicitly and then resolve this issue by assuming we have blackbox algorithms that operate over the implicit matrix.

We define our stream such that each element in the stream d_k is a pair of values (i, j):

▶ **Definition 4** (Stream). Let m, n be positive integers. A **stream** $D = D(m, n)$ is a sequence of length m, $d_1, d_2, \dots, d_m$, where each entry is a pair of values in $\{1, \dots, n\}$.

Let $g : \mathbb{R} \to \mathbb{R}$ be a non-negative, subadditive, and even function where $g(0) = 0$. Frequently, we will need to discuss a matrix where g has been applied to every entry. We use the notations from [12] which are in turn based on notations from [13].

▶ **Definition 5** (Hadamard Function). Given a matrix A of dimensions $n \times n$, a **Hadamard function** g takes as input the matrix A and is applied entrywise to every entry of the matrix. The output is the matrix $g[A]$. Further, we note that the L_1 norm of $g[A]$ is equivalent to the value we aim to approximate, $\|g[A]\|_1 = \sum_{i=1}^{n} \sum_{j=1}^{n} g(a_{ij})$.

We frequently use hash functions in our analysis, we now specify some notation. We sometimes express a hash function H over a domain of size n as a vector of values $(h_1, h_2, ..., h_n)$. Multiplication of two hash functions H^a, H^b is given by the Hadamard product, denoted $H' = HAD(H^a, H^b) = H^a H^b$, where multiplication is performed entrywise so that $(h'_1 = h^a_1 h^b_1, ..., h'_n = h^a_n h^b_n)$.

We now define two additional matrices. All matrices in our definitions are of size $n \times n$, and all vectors are of size $1 \times n$. We denote by $[n]$ the set $\{1, \dots, n\}$.

▶ **Definition 6** (Sampling Identity Matrix)**.** Given a hash function $H : [n] \to \{0,1\}$, let $h_i = H(i)$. The **Sampling Identity Matrix** I_H with entries b_{ij} is defined as:

$$I_H = \begin{cases} b_{ii} = h_i \\ b_{ij} = 0 \text{ for } i \neq j. \end{cases}$$

That is, the diagonal of I_H corresponds to the values of H. When we multiply matrix I_H by A, each row of $I_H A$ is either the zero vector (corresponding to $h_i = 0$) or the original row i in A (corresponding to $h_i = 1$). We use the term "sampling" due to the fact that the hash functions we use throughout this paper are random, and hence which rows remain untouched is random. The same observations apply to columns when considering the matrix AI_H.

▶ **Definition 7** (Row Aggregation Vector)**.** A **Row Aggregation Vector** J is a $1 \times n$ vector with all entries equal to 1.

Thus, JA yields a vector V where each value v_j is $\sum_{i=1}^{n} a_{ij}$.

▶ **Blackbox Algorithm 1** ($(1 \pm \epsilon')$-Approximation of g on an aggregated matrix)**.**
Input: *Matrix A, and hash function H.*
Output: $(1 \pm \epsilon')$*-Approximation of* $\|g[JI_H A]\|_1$ *with probability* $(1 - \delta_1)$.

The space Blackbox Algorithm 1 (BA1) uses is referred to as $SPACE1(n, \delta_1, \epsilon')$ in our analysis.

▶ **Blackbox Algorithm 2** ($r(n)$-Approximation of $\|g[I_H A]\|_1$)**.**
Input: *Matrix A, and hash function H.*
Output: $r(n)$*-Approximation of* $\|g[I_H A]\|_1$ *with probability* $(1 - \delta_2)$.

The space Blackbox Algorithm 2 ($BA2$) uses is referred to as $SPACE2(n, \delta_2)$ in our analysis.

▶ **Definition 8** (Complement Hash Function)**.** For a hash function $H : [n] \to \{0,1\}$, define the **Complement Hash Function** $\bar{H} : [n] \to \{0,1\}$ as $\bar{H}(i) = 1$ if and only if $H(i) = 0$.

▶ **Definition 9** (Threshold Functions)**.** We define two **Threshold Functions**, which we denote by $\rho(n,\epsilon) = O(\frac{r^4(n)}{\epsilon})$ and $\tau(n,\epsilon) = O(\frac{r^2(n)}{\epsilon})$.

▶ **Definition 10** (Weight of a Row)**.** The **weight** of row i in matrix A is given by $u_{A,i} = \sum_{j=1}^{n} a_{ij}$.

▶ **Definition 11** (α-Heavy Rows)**.** Row i is α**-heavy** for $0 < \alpha < 1$ if $u_{A,i} > \alpha \|A\|_1$.

▶ **Definition 12** (Key Row)**.** We say row i is a **Key Row** if: $u_{A,i} > \rho(n,\epsilon)(\|A\|_1 - u_{A,i})$.

While Definition 11 and Definition 12 are similar, we define them for convenience, as our algorithm works by first finding key rows and then building on top of this to find α-heavy rows. We note that, as long as $\rho(n,\epsilon) \geq 1$, a matrix can have at most one key row (since any matrix can have at most $\frac{1}{\alpha}$ α-heavy rows, and a key row is α-heavy for $\alpha = \frac{\rho(n,\epsilon)}{1+\rho(n,\epsilon)}$).

3 Subadditive Approximations

In this section we show that a $(1 \pm \epsilon)$-approximation of even, subadditive, non-negative Hadamard functions which are zero at the origin are preserved under row or column aggregations in the presence of sufficiently heavy rows or columns.

▶ **Theorem 13.** *Let B be an $n \times n$ matrix and let $\epsilon \in (0,1)$ be a parameter. Recall that J is a row vector with all entries equal to 1. Let g be any even, subadditive, non-negative Hadamard function which satisfies $g(0) = 0$. Denote $u_i = \sum_{j=1}^{n} g(b_{ij})$, and thus $\|g[B]\|_1 = \sum_{i=1}^{n} u_i$. If there is a row h such that $u_h \geq (1 - \frac{\epsilon}{2})\|g[B]\|_1$, then $|u_h - \|g[JB]\|_1| \leq \epsilon\|g[JB]\|_1$.*

Proof. Denote $V = JB$. Without loss of generality assume u_1 is the row such that $u_1 \geq (1 - \frac{\epsilon}{2})\|g[B]\|_1$. By subadditivity of g: $\|g[V]\|_1 \leq \|g[B]\|_1 \leq \frac{1}{1-\frac{\epsilon}{2}}u_1 \leq (1+\epsilon)u_1$. Further, we have $b_{1j} = (\sum_{i=1}^{n} b_{ij} - \sum_{i=2}^{n} b_{ij})$. Since g is even and subadditive, and such functions are non-negative, we have $g(b_{1j}) \leq g\left(\sum_{i=1}^{n} b_{ij}\right) + \sum_{i=2}^{n} g(b_{ij})$. Rearranging and summing over j, we get: $\sum_{j=1}^{n} g\left(\sum_{i=1}^{n} b_{ij}\right) \geq \sum_{j=1}^{n}\left(g(b_{1,j}) - \sum_{i=2}^{n} g(b_{ij})\right)$.

Therefore:

$$\|g[V]\|_1 = \sum_{j=1}^{n} g\left(\sum_{i=1}^{n} b_{ij}\right) \geq \sum_{j=1}^{n}\left(g(b_{1,j}) - \left(\sum_{i=2}^{n} g(b_{ij})\right)\right) = u_1 - (\|g[B]\|_1 - u_1).$$

Finally:

$$\begin{aligned}\|g[V]\|_1 \geq u_1 - (\|g[B]\|_1 - u_1) = 2u_1 - \|g[B]\|_1 &\geq u_1\left(2 - \frac{1}{1-\frac{\epsilon}{2}}\right)\\ &= u_1\frac{1-\epsilon}{1-\frac{\epsilon}{2}} \geq u_1(1-\epsilon).\end{aligned}$$ ◀

4 Algorithm for Finding Key Rows

▶ **Definition 14** (Algorithm for Finding Key Rows).
Input: Matrix A and Sampling Identity Matrix I_H generated from hash function H.
Output: Pair (a, b), where the following holds for a, b, and the matrix $W = I_H A$:

1. The pair is either $(a, b) = (-1, 0)$ or $(a, b) = (i, \tilde{u}_{W,i})$. Here, $\tilde{u}_{W,i}$ is a $(1 \pm \epsilon)$-approximation to $u_{W,i}$ and the index i is the correct corresponding row.
2. If there is a key row i_0 for the matrix W, then $a = i_0$.

Before describing the algorithm and proving its correctness, we prove the following useful lemma in Appendix A.

▶ **Lemma 15.** *Let $U = (u_1, \ldots, u_n)$ be a vector with non-negative entries of dimension n and let H' be a pairwise independent hash function where $H' : [n] \to \{0,1\}$ and $P[H'(i) = 1] = P[H'(i) = 0] = \frac{1}{2}$. Denote by $\bar{H}'$ the hash function defined by $\bar{H}'(i) = 1 - H'(i)$. Let $X = \sum_i H'(i)u_i$ and $Y = \sum_i \bar{H}'(i)u_i$. If there is no $\frac{1}{16}$-heavy element with respect to U, then:*

$$\Pr\left[\left(X \leq \frac{1}{4} \cdot \|U\|_1\right) \cup \left(Y \leq \frac{1}{4} \cdot \|U\|_1\right)\right] \leq \frac{1}{4}.$$

▶ **Theorem 16.** *If there exist two blackbox algorithms as specified in Blackbox Algorithms 1 and 2, then there exists an algorithm that satisfies the requirements in Definition 14 with high probability.*

Proof. We will prove that Algorithm 1 fits the description of Definition 14. Using standard methods such as in [10], we have a loop that runs in parallel $O(\log(n))$ times so that we can find the index of a heavy element and return it, if there is one. To prove this theorem, we consider the following three exhaustive and disjoint cases regarding the matrix $g[I_H A]$ (recall that $H : [n] \to \{0,1\}$):

Algorithm 1 Algorithm Find-Key-Row

```
The algorithm takes as input a matrix $A$ and a hash function $H : [n] \to \{0,1\}$
  for $\ell = 1$ to $N = O(\log n)$ do
    Generate a pairwise independent, uniform hash function $H_\ell : [n] \to \{0,1\}$
    Let $T_1 = HAD(H, H_\ell)$, $T_0 = HAD(H, \bar{H}_\ell)$
    Let $y_1 = BA2(A, T_1)$, $y_0 = BA2(A, T_0)$ ($BA2$ is run with constant failure probability $\delta_2$)
    if $y_0 \geq \tau(n, \epsilon) \cdot y_1$ then
      $b_\ell = 0$
    else if $y_1 \geq \tau(n, \epsilon) \cdot y_0$ then
      $b_\ell = 1$
    else
      $b_\ell = 2$
  if $|\{\ell : b_\ell = 2\}| \geq \frac{2}{5} \cdot N$ then
    Return $(-1, 0)$
  else
    if there is a row $i$ such that $i$ satisfies $|\{\ell : H_\ell(i) = b_\ell\}| \geq \frac{3}{4} \cdot N$ then
      Return $(i, BA1(A, H))$ ($BA1$ is run with $\epsilon' = \frac{\epsilon}{2}$ and $\delta_1$ is set to be inverse polylogarithmic)
    else
      Return $(-1, 0)$
```

1. The matrix has a key row (note that a matrix always has at most one key row).
2. The matrix has no α-heavy row for $\alpha = 1 - \frac{\epsilon}{8}$.
3. The matrix has an α-heavy row for $\alpha = 1 - \frac{\epsilon}{8}$, but there is no key row.

We prove that the algorithm is correct in each case in Lemmas 22, 23, and 24, respectively. These proofs can be found in Appendix B. ◀

With the proofs of these three cases, we are done proving that Algorithm 1 performs correctly. We now analyze the space bound for Algorithm 1.

▶ **Lemma 17.** *Algorithm 1 uses* $O\left(SPACE1(n, \delta_1, \frac{\epsilon}{2}) + \log(n)(\log^2(n) + SPACE2(n, \delta_2))\right)$ *bits of memory, where* δ_1 *is inverse polylogarithmic and* δ_2 *is a constant.*

Proof. Note that, in order for our algorithm to succeed, we run $BA1$ with an error parameter of $\epsilon' = \frac{\epsilon}{2}$ and a failure probability parameter δ_1 which is inverse polylogarithmic. Moreover, we run $BA2$ with a constant failure probability. We also require a number of random bits bounded by $O(\log^2(n))$ for generating each hash function H_ℓ, as well as the space required to run $BA2$ in each iteration of the loop. Since there are $O(\log n)$ parallel iterations, this gives the lemma. ◀

4.1 Algorithm for Finding All α-Heavy Rows

Algorithm 1 only guarantees that we return key rows. Given a matrix A, we now show that this algorithm can be used as a subroutine to find all α-heavy rows i with respect to the matrix $g[A]$ with high probability, along with a $(1 \pm \epsilon)$-approximation to the row weights $u_{g[A],i}$ for all i. In order to do this, we apply an additional hash function $H : [n] \to [\tau]$ which essentially maps rows of the matrix to some number of buckets τ (i.e., each bucket

Algorithm 2 Algorithm Find-Heavy-Rows

```
The algorithm takes as input a matrix $A$ and a value $0 < \alpha < 1$
  Generate a pairwise independent hash function $H : [n] \to [\tau]$, where $\tau = O\left(\frac{\rho(n,\epsilon)\log(n)}{\alpha^2}\right)$
  for $k = 1$ to $\tau$ do
    Let $H_k : [n] \to \{0, 1\}$ be the function defined by $H_k(i) = 1 \iff H(i) = k$
    Let $C_k$ = Find-Key-Row$(A, H_k)$
  Return $\{C_k : C_k \neq (-1, 0)\}$
```

corresponds to a set of sampled rows based on H), and then run Algorithm 1 for each bucket. The intuition for why the algorithm works is that any α-heavy row i in the original matrix A is likely to be a key row for the matrix in the corresponding bucket to which row i is mapped. Note that, eventually, we find α-heavy rows for $\alpha = \frac{\epsilon^2}{\log^3 n}$. The algorithm sets $\tau = O\left(\frac{\rho(n,\epsilon)\log(n)}{\alpha^2}\right)$ and is given below.

▶ **Theorem 18.** *Algorithm 2 outputs a set of pairs $Q = \{(i_1, a_1), \dots, (i_t, a_t)\}$ for $t \le \tau$ which satisfies the following properties, except with probability $\frac{1}{\log n}$:*
1. *$\forall j \in [t]$: $(1-\epsilon)u_{g[A],i_j} \le a_j \le (1+\epsilon)u_{g[A],i_j}$.*
2. *$\forall i \in [n]$: If row i is α-heavy with respect to the matrix $g[A]$, then $\exists j \in [t]$ such that $i_j = i$ (for any $0 < \alpha < 1$).*

Proof. First, the number of pairs output by Algorithm 2 is at most the number of buckets, which equals τ. Now, the first property is true due to the fact that Algorithm 1 has a high success probability. In particular, as long as the failure probability is at most $\frac{1}{\tau \cdot \log^c(n)}$ for some constant c (which we ensure), then by the union bound the probability that there exists a pair $(i_j, a_j) \in Q$ such that a_j is not a $(1 \pm \epsilon)$-approximation to $u_{g[A],i_j}$ is at most inverse polylogarithmic.

Now, to ensure the second item, we need to argue that every α-heavy row gets mapped to its own bucket with high probability, since if there is a collision the algorithm cannot find all α-heavy rows. Moreover, we must argue that for each α-heavy row i with respect to the matrix $g[A]$, if i is mapped to bucket k by H, then row i is actually a key row in the corresponding sampled matrix $g[A_k]$ (for ease of notation, we write A_k to denote the matrix H_kA_k). More formally, suppose row i is α-heavy. Then the algorithm must guarantee with high probability that, if $H(i) = k$, then row i is a key row in the matrix $g[A_k]$. If we prove these two properties, then the theorem holds (since Algorithm 1 outputs a key row with high probability, if there is one).

Observe that there must be at most $\frac{1}{\alpha}$ rows which are α-heavy. In particular, let R be the set of α-heavy rows, and assume towards a contradiction that $|R| > \frac{1}{\alpha}$. Then we have:

$$\|g[A]\|_1 \ge \sum_{i \in R} u_{g[A],i} \ge \sum_{i \in R} \alpha \|g[A]\|_1 = \alpha \cdot \|g[A]\|_1 \cdot |R| > \|g[A]\|_1,$$

which is a contradiction. Hence, we seek to upper bound the probability of a collision when throwing $\frac{1}{\alpha}$ balls into τ bins. By a Birthday paradox argument, this happens with probability at most $\frac{1}{2 \cdot \tau \cdot \alpha^2}$, which can be upper bounded as follows:

$$\frac{1}{2\tau\alpha^2} \le \frac{\alpha^2}{2\alpha^2 \rho(n,\epsilon)\log(n)} = \frac{1}{2\rho(n,\epsilon)\log(n)} \le \frac{\epsilon}{2r^4(n)\log(n)},$$

which is inverse polylogarithmically small.

Now, we argue that every α-heavy row i for the matrix $g[A]$ is mapped to a sampled matrix such that i is a key row in the sampled matrix with high probability. In particular, suppose $H(i) = k$, implying that row i is mapped to bucket k. For $\ell \neq i$, let X_ℓ be the indicator random variable which is 1 if and only if row ℓ is mapped to the same bucket as i, namely $H(\ell) = k$ (i.e., $X_\ell = 1$ means the sampled matrix $g[A_k]$ contains row i and row ℓ). If row i is not a key row for the matrix $g[A_k]$, this means that $u_{g[A_k],i} \leq \rho(n,\epsilon)(\|g[A_k]\|_1 - u_{g[A_k],i})$. Observe that, if row i is mapped to bucket k, then we have $u_{g[A_k],i} = u_{g[A],i}$. Hence, the the probability that row i is not a key row for the sampled matrix $g[A_k]$ (assuming row i is mapped to bucket k) can be expressed as $\Pr[u_{g[A],i} \leq \rho(n,\epsilon)(\|g[A_k]\|_1 - u_{g[A],i}) | H(i) = k]$. By pairwise independence of H, and by Markov's inequality, we can write:

$$\begin{aligned}
&\Pr\left[u_{g[A],i} \leq \rho(n,\epsilon)(\|g[A_k]\|_1 - u_{g[A],i}) \;\middle|\; H(i) = k\right] \\
&\quad = \Pr\left[u_{g[A],i} \leq \rho(n,\epsilon)\sum_{\ell \neq i} u_{g[A],\ell}X_\ell \;\middle|\; H(i) = k\right] \\
&\quad = \Pr\left[u_{g[A],i} \leq \rho(n,\epsilon)\sum_{\ell \neq i} u_{g[A],\ell}X_\ell\right] \\
&\quad = \Pr\left[\sum_{\ell \neq i} u_{g[A],\ell}X_\ell \geq \frac{u_{g[A],i}}{\rho(n,\epsilon)}\right] \leq \frac{\rho(n,\epsilon)\mathbb{E}\left[\sum_{\ell \neq i} u_{g[A],\ell}X_\ell\right]}{u_{g[A],i}} \\
&\quad = \frac{\rho(n,\epsilon)\sum_{\ell \neq i} u_{g[A],\ell}}{\tau \cdot u_{g[A],i}} \leq \frac{\rho(n,\epsilon)\|g[A]\|_1}{\alpha\tau\|g[A]\|_1} = \frac{\alpha^2\rho(n,\epsilon)}{4\alpha \cdot \rho(n,\epsilon)\log(n)} \leq \frac{\alpha}{4\log(n)}.
\end{aligned}$$

Here, we choose $\tau = \frac{4\rho(n,\epsilon)\log(n)}{\alpha^2}$, and get that the probability that a particular α-heavy row i is not a key row in its corresponding sampled matrix is at most $\frac{\alpha}{4\log(n)}$. Since there are at most $\frac{1}{\alpha}$ rows which are α-heavy, by the union bound the probability that there exists an α-heavy row that is not a key row in its sampled matrix is at most $\frac{1}{4\log(n)}$.

Thus, in all, the probability that at least one bad event happens (i.e., there exists a pair (i_j, a_j) such that a_j is not a good approximation to $u_{g[A],i_j}$, there is a collision between α-heavy rows, or an α-heavy row is not a key row in its corresponding sampled matrix) is at most $\frac{1}{\log(n)}$. This gives the theorem. ◀

4.2 Sum from α-Heavy Rows

We now have an algorithm that is able to find all α-heavy rows for $\alpha = \frac{\epsilon^2}{\log^3 n}$, except with probability $\frac{1}{\log n}$. In the language of [11], by Theorem 18, our α-heavy rows algorithm outputs an (α,ϵ)-cover with respect to the vector $(u_{g[A],1}, u_{g[A],2}, \ldots, u_{g[A],n})$ except with probability $\frac{1}{\log n}$, where $\epsilon > 0$ and $\alpha > 0$. Hence, we can apply the Recursive Sum algorithm from [11] (see Appendix C for the formal definition of an (α,ϵ)-cover, along with the Recursive Sum algorithm) to get a $(1 \pm \epsilon)$-approximation of $\|g[A]\|_1$. Note that the Recursive Sum algorithm needs $\alpha = \frac{\epsilon^2}{\log^3 n}$ and a failure probability of at most $\frac{1}{\log n}$, which we provide. Hence, we get the following theorem.

▶ **Theorem 19.** *The Recursive Sum Algorithm, using Algorithm 2 as a subroutine, returns a $(1 \pm \epsilon)$-approximation of $\|g[A]\|_1$.*

4.3 Space Bounds

▶ **Lemma 20.** *Recursive Sum, using Algorithm 2 as a subroutine as described in Section 4.2, uses the following amount of memory, where $\epsilon' = \frac{\epsilon}{2}$, δ_1 is inverse polylogarithmic, and δ_2 is a small constant:*

$$O\left(\frac{r^4(n)}{\epsilon^5}(\log^{10}(n) + \log^8(n)SPACE1(n, \delta_1, \epsilon') + \log^9(n)SPACE2(n, \delta_2))\right).$$

Proof. The final algorithm uses the space bound from Lemma 17, multiplied by $\tau = O\left(\frac{\rho(n,\epsilon)\log(n)}{\alpha^2}\right)$, where $\alpha = \frac{\epsilon^2}{\phi^3}$, $\phi = O(\log n)$, and $\rho(n, \epsilon) = O(\frac{r^4(n)}{\epsilon})$. This gives $\tau = \frac{1}{\epsilon^5}r^4(n)\log^7(n)$ to account for the splitting required to find α-heavy rows in Section 4.1. Finally, a multiplicative cost of $\log(n)$ is needed for the Recursive Sum algorithm, giving the final bound. ◀

5 Applications

We now apply our algorithm to the problem of determining the L_1 distance between the joint and product of the marginal distributions as described in Problem 1.

Space Bounds for Determining L_1 Independence

Given an $n \times n$ matrix A with entries $a_{ij} = \frac{f_{ij}}{m} - \frac{p_i q_j}{m}$, we have provided a method to approximate the value $\|g[A]\|_1$:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} g\left(\frac{f_{ij}}{m} - \frac{p_i q_j}{m}\right).$$

Let g be the L_1 distance, namely $g(x) = |x|$ (hence, the $(i, j)^{th}$ entry in $g[A]$ is given by $|\frac{f_{ij}}{m} - \frac{p_i q_j}{m}|$). we now state explicitly which blackbox algorithms we use:

- Let Blackbox Algorithm 1 ($BA1$) be the $(1 \pm \epsilon')$-approximation of L_1 for vectors from [14]. The space of this algorithm is upper bounded by the number of random bits required and uses $O(\log(\frac{nm}{\delta_1\epsilon'})\log(\frac{m}{\delta_1\epsilon'})\log(\frac{1}{\delta_1})\epsilon'^{-2})$ bits of memory.
- Let Blackbox Algorithm 2 ($BA2$) be the $r(n)$-approximation, using the L_1 sketch of the distance between the joint and product of the marginal distributions from [15]. This algorithm does not have a precise polylogarithmic bound provided, but we compute that it is upper bounded by the random bits required to generate the Cauchy random variables similarly to $BA1$ (which are generated in parallel $O(\log\frac{1}{\delta_2})$ times). This algorithm requires $O(\log(\frac{nm}{\delta_1\epsilon'})\log(\frac{m}{\delta_1\epsilon'})\log(\frac{1}{\delta_1})\log(\frac{1}{\delta_2})\epsilon'^{-2})$ bits of memory. Note that $BA2$ does not depend on ϵ', δ_1, but we are stating a loose upper bound.

These two algorithms match the definitions given in Section 2, and thus we are able to give a bound of $O(\frac{1}{\epsilon^7}\log^{14}(n)\log^2(\frac{nm}{\epsilon}))$ on the space our algorithm requires (recall that we set $\epsilon' = \theta(\epsilon)$, δ_2 to be some small constant, and δ_1 to be inverse polylogarithmically small). We can improve this slightly as follows.

▶ **Corollary 21.** *Due to the nature of the truncated Cauchy distribution (see [15]), we can further improve our space bound to $O\left(\frac{1}{\epsilon^7}\log^{12}(n)\log^2(\frac{nm}{\epsilon})\right)$.*

Proof. Due to the constant lower bound on the approximation of L_1, instead of $\frac{1}{r^2(n)} \le \|g[W]\|_1 \le r^2(n)$, we get $C \le \|g[W]\|_1 \le \log^2(n)$ for some constant C. As the space cost from dividing the matrix into submatrices as shown in Section 4.1 directly depends on these bounds, we only pay an $O(r^2(n))$ multiplicative factor instead of an $O(r^4(n))$ multiplicative factor and achieve a bound of $O\left(\frac{1}{\epsilon^7}\log^{12}(n)\log^2(\frac{nm}{\epsilon})\right)$. ◀

References

1 Noga Alon, Alexandr Andoni, Tali Kaufman, Kevin Matulef, Ronitt Rubinfeld, and Ning Xie. Testing k-wise and almost k-wise independence. In *Proceedings of the 39th annual ACM Symposium on Theory of Computing*, 2007.

2 Noga Alon, Oded Goldreich, and Yishay Mansour. Almost k-wise independence versus k-wise independence. *Information Processing Letters*, 88(3):107–110, 2003.

3 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the 28th annual ACM Symposium on Theory of Computing*, 1996.

4 Tuğkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *Proceedings of the 42nd annual IEEE Symposium on Foundations of Computer Science*, 2001.

5 Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing that distributions are close. In *Proceedings of the 41st annual IEEE Symposium on Foundations of Computer Science*, 2000.

6 Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM*, 60(1):4, 2013.

7 Tuğkan Batu, Ravi Kumar, and Ronitt Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of the 36th annual ACM Symposium on Theory of Computing*, 2004.

8 Vladimir Braverman, Kai-Min Chung, Zhenming Liu, Michael Mitzenmacher, and Rafail Ostrovsky. AMS Without 4-Wise Independence on Product Domains. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science*, 2010.

9 Vladimir Braverman and Rafail Ostrovsky. Measuring independence of datasets. In *Proceedings of the 42nd annual ACM Symposium on Theory of Computing*, 2010.

10 Vladimir Braverman and Rafail Ostrovsky. Zero-one frequency laws. In *Proceedings of the 42nd annual ACM Symposium on Theory of Computing*, 2010.

11 Vladimir Braverman and Rafail Ostrovsky. Generalizing the layering method of Indyk and Woodruff: Recursive sketches for frequency-based vectors on streams. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*. Springer, 2013.

12 Dominique Guillot, Apoorva Khare, and Bala Rajaratnam. Complete characterization of hadamard powers preserving loewner positivity, monotonicity, and convexity. *Journal of Mathematical Analysis and Applications*, 425(1):489–507, 2015.

13 Roger A. Horn and Charles R. Johnson. Topics in matrix analysis. *Cambridge University Press, Cambridge*, 1991.

14 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, 2006.

15 Piotr Indyk and Andrew McGregor. Declaring independence via the sketching of sketches. In *Proceedings of the 19th annual ACM-SIAM Symposium on Discrete Algorithms*, 2008.

16 Ralph Kimball and Joe Caserta. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. John Wiley & Sons, 2004.

17 Erich L. Lehmann and Joseph P. Romano. *Testing statistical hypotheses*. Springer Science & Business Media, 2006.

18 Andrew McGregor and Hoa T. Vu. Evaluating bayesian networks via data streams. In *Proceedings of the 21st International Computing and Combinatorics Conference*, 2015.

19 Viswanath Poosala and Yannis E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, 1997.

20 Ronitt Rubinfeld and Rocco A. Servedio. Testing monotone high-dimensional distributions. In *Proceedings of the 37th annual ACM Symposium on Theory of Computing*, 2005.

21 Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM*, 50(2):196–249, 2003.

22 David P. Woodruff. Sketching as a tool for numerical linear algebra. *CoRR*, abs/1411.4357, 2014. URL: `http://arxiv.org/abs/1411.4357`.

A Proof of Lemma 15

Proof. Note that we always have the equality $X+Y=\sum_i H'(i)u_i+\bar{H}'(i)u_i=\sum_i H'(i)u_i+(1-H'(i))u_i=\|U\|_1$, and moreover $\mathbb{E}[X]=\sum_i u_i\mathbb{E}[H'(i)]=\frac{1}{2}\cdot\|U\|_1$. Also, observe that

$$\begin{aligned}Var[X]&=\mathbb{E}[X^2]-(\mathbb{E}[X])^2\\&=\sum_i\mathbb{E}[(H'(i))^2]u_i^2+\sum_{i\neq j}\mathbb{E}[H'(i)H'(j)]u_iu_j-\frac{1}{4}\cdot\|U\|_1^2\\&=\frac{1}{2}\sum_i u_i^2+\frac{1}{4}\sum_{i\neq j}u_iu_j-\frac{1}{4}\left(\sum_i u_i^2+\sum_{i\neq j}u_iu_j\right)=\frac{1}{4}\sum_i u_i^2.\end{aligned}$$

Using the fact that there is no $\frac{1}{16}$-heavy element with respect to U, which implies that $u_i\le\frac{1}{16}\cdot\|U\|_1$ for all i, we have:

$$Var[X]=\frac{1}{4}\sum_i u_i^2\le\frac{\|U\|_1}{64}\sum_i u_i=\frac{\|U\|_1^2}{64}.$$

Now we can apply Chebyshev's inequality to obtain:

$$\begin{aligned}\Pr\left[\left(X\le\frac{1}{4}\cdot\|U\|_1\right)\cup\left(Y\le\frac{1}{4}\cdot\|U\|_1\right)\right]&=\Pr\left[|X-\mathbb{E}[X]|\ge\frac{\|U\|_1}{4}\right]\\&\le\frac{16\cdot Var[X]}{\|U\|_1^2}\le\frac{16\cdot\|U\|_1^2}{64\cdot\|U\|_1^2}=\frac{1}{4}.\end{aligned}$$ ◀

B Proof of Correctness of Algorithm 1

Throughout the lemmas, we imagine that the hash function $H:[n]\to\{0,1\}$ is fixed, and hence the matrix $g[I_HA]$ is fixed. All randomness is taken over the pairwise independent hash functions H_ℓ that are generated in parallel, along with both blackbox algorithms.

To ease the notation, we define

$$W=I_HA,\ W_1=I_{T_1}A,\text{ and }W_0=I_{T_2}A$$

(recall the notation from Algorithm 1 that $T_1=HAD(H,H_\ell)$ and $T_0=HAD(H,\bar{H}_\ell)$). Finally, for each row i in the matrix $g[W]$, we define the shorthand notation $u_i=u_{g[W],i}$.

▶ **Lemma 22.** *If the matrix $g[I_HA]$ has a key row, Algorithm 1 correctly returns the index of the row and a $(1\pm\epsilon)$-approximation of the weight of the key row except with inverse polylogarithmic probability.*

Proof. Suppose the matrix $g[I_HA]$ has a key row, and let i_0 be the index of this row. We prove that we return a good approximation of $u_{g[W],i_0}$ with high probability. In particular, we first argue that, for a fixed iteration ℓ of the loop, we have the property that b_ℓ equals

$H_\ell(i_0)$, and moreover this holds with high probability. We assume without loss of generality that $H_\ell(i_0) = 1$ (the case when $H_\ell(i_0) = 0$ is symmetric). In particular, this implies that the key row i_0 appears in the matrix $g[W_1]$.

By definition of $BA2$, the following holds for $y_1 = BA2(A, T_1)$ and $y_0 = BA2(A, T_0)$, except with probability $2\delta_2$ (where δ_2 is the failure probability of $BA2$):

$$y_1 \geq \frac{\|g[W_1]\|_1}{r(n)} \text{ and } y_0 \leq \|g[W_0]\|_1 r(n).$$

We have the following set of inequalities:

$$\|g[W_1]\|_1 \geq u_{i_0} > \rho(n,\epsilon)(\|g[W]\|_1 - u_{i_0}) \geq \rho(n,\epsilon)\|g[W_0]\|_1,$$

where the first inequality follows since g is non-negative and the key row i_0 appears in the matrix $g[W_1]$ (and hence the L_1 norm of $g[W_1]$ is at least u_{i_0} since it includes the row i_0), the second inequality follows by definition of i_0 being a key row for the matrix W, and the last inequality follows since the entries in row i_0 of the matrix W_0 are all zero (as $H_\ell(i_0) = 1$) and the remaining rows of W_0 are sampled from W, along with the facts that g is non-negative and $g(0) = 0$.

Substituting for $\rho(n,\epsilon)$, and using the fact that y_1 and y_0 are good approximations for $\|g[W_1]\|_1$ and $\|g[W_0]\|_1$ (respectively), except with probability $2\delta_2$, we get:

$$y_1 \geq \frac{\|g[W_1]\|_1}{r(n)} > \frac{\rho(n,\epsilon)}{r(n)} \cdot \|g[W_0]\|_1 \geq \frac{\rho(n,\epsilon)}{r^2(n)} \cdot y_0 \geq \tau(n,\epsilon) \cdot y_0.$$

Thus, in this iteration of the loop we have $b_\ell = 1$ except with probability $2\delta_2$ (in the case that $H_\ell(i_0) = 0$, it is easy to verify by a similar argument that $y_0 \geq \tau(n,\epsilon) \cdot y_1$, and hence we have $b_\ell = 0$). Hence, for the row i_0, we have the property that $b_\ell = H_\ell(i_0)$ for a fixed ℓ, except with probability $2\delta_2$. By the Chernoff bound, as long as δ_2 is a sufficiently small constant, we have $b_\ell = H_\ell(i_0)$ for at least a $\frac{3}{4}$-fraction of iterations ℓ, except with inverse polynomial probability. The only issue to consider is the case that there exists another row $i \neq i_0$ with the same property, namely $b_\ell = H_\ell(i)$ for a large fraction of iterations ℓ. However, if $b_\ell = H_\ell(i)$, it must be that at least one of y_1, y_0 is a bad approximation or $H_\ell(i) = H_\ell(i_0)$, which happens with probability at most $2\delta_2 + \frac{1}{2}$. Therefore, by the Chernoff bound, the probability that this happens for at least a $\frac{3}{4}$-fraction of iterations ℓ is at most $\frac{1}{2^{O(\log n)}}$, which is inverse polynomially small. By applying the union bound, the probability that there exists such a row is at most $\frac{n-1}{2^{O(\log n)}}$, which is at most an inverse polynomial. Hence, in this case, the algorithm returns $(i_0, BA1(A,H))$ except with inverse polynomial probability.

We now argue that $\tilde{u}_{g[W],i_o} = BA1(A,H)$ is a $(1 \pm \epsilon)$-approximation of $u_{g[W],i_0}$, except with inverse polylogarithmic probability. By definition of $BA1$, which we run with an error parameter of $\epsilon' = \frac{\epsilon}{2}$, it returns a $\left(1 \pm \frac{\epsilon}{2}\right)$-approximation of $\|g[JW]\|_1$ except with inverse polylogarithmic probability, where $W = I_H A$. Moreover, since i_0 is a key row, we have:

$$u_{i_0} > \rho(n,\epsilon)(\|g[W]\|_1 - u_{i_0}) \Rightarrow u_{i_0} > \frac{\rho(n,\epsilon)\|g[W]\|_1}{1 + \rho(n,\epsilon)} \geq \left(1 - \frac{\epsilon}{8}\right)\|g[W]\|_1,$$

where the last inequality follows as long as $r^4(n) \geq 8 - \epsilon$. This implies that i_0 is $\left(1 - \frac{\epsilon}{8}\right)$-heavy with respect to the matrix $g[W]$, and hence we can apply Theorem 13 to get that:

$$\begin{aligned}(1+\epsilon)u_{i_0} \geq \frac{(1+\frac{\epsilon}{2})}{(1-\frac{\epsilon}{4})}u_{i_0} \geq \left(1+\frac{\epsilon}{2}\right)\|g[JW]\|_1 \geq \tilde{u}_{g[W],i_0} \\ \geq \left(1-\frac{\epsilon}{2}\right)\|g[JW]\|_1 \geq \frac{(1-\frac{\epsilon}{2})}{(1+\frac{\epsilon}{4})}u_{i_0} \geq (1-\epsilon)u_{i_0}.\end{aligned}$$

The first inequality holds for any $0 < \epsilon \le 1$, the second inequality holds by Theorem 13, the third inequality holds since $\tilde{u}_{g[W],i_0}$ is a $\left(1 \pm \frac{\epsilon}{2}\right)$-approximation of $\|g[JW]\|_1$, and the rest hold for similar reasons. Hence, our algorithm returns a good approximation as long as $BA1$ succeeds. Noting that this happens except with inverse polylogarithmic probability gives the lemma. ◀

▶ **Lemma 23.** *If the input matrix has no α-heavy row, where $\alpha = 1 - \frac{\epsilon}{8}$, then with high probability Algorithm 1 correctly returns* $(-1, 0)$.

Proof. In this case, we have no α-heavy row for $\alpha = 1 - \frac{\epsilon}{8}$, which implies that $u_i \le \alpha\|g[W]\|_1 = \left(1 - \frac{\epsilon}{8}\right)\|g[W]\|_1$ for each row i in the matrix $g[W]$. In this case, we show the probability that Algorithm 1 returns a false positive is small. That is, with high probability, in each iteration ℓ of the loop the algorithm sets $b_\ell = 2$, and hence it returns $(-1, 0)$. We split this case into three additional disjoint and exhaustive subcases, defined as follows:

1. For each row i, we have $u_i \le \frac{1}{16}\|g[W]\|_1$.
2. There exists a row i with $u_i > \frac{1}{16}\|g[W]\|_1$ and $\forall j \ne i$ we have $u_j \le \frac{\epsilon}{128}u_i$.
3. There exist two distinct rows i, j where $u_i > \frac{1}{16}\|g[W]\|_1$ and $u_j > \frac{\epsilon}{128}u_i$.

We define $X = \sum_i h_i^\ell u_i$ and $Y = \sum_i \bar{h}_i^\ell u_i$, where $h_i^\ell = H_\ell(i)$ and $\bar{h}_i^\ell = \bar{H}_\ell(i)$. Hence, we have $X = \|g[W_1]\|_1$ and $Y = \|g[W_0]\|_1$, and moreover $X + Y = \|g[W]\|_1$ (recall that $g[W_1] = g[I_{T_1}A]$ and $g[W_0] = g[I_{T_0}A]$).

In the first subcase, where there is no $\frac{1}{16}$-heavy row, we can apply Lemma 15 to the vector $(u_1, \ldots, u_n)$ to get that:

$$\Pr\left[\left(X \le \frac{\|g[W]\|_1}{4}\right) \cup \left(Y \le \frac{\|g[W]\|_1}{4}\right)\right] \le \frac{1}{4}.$$

By definition of $BA2$, the following holds for $y_1 = BA2(A, T_1)$ and $y_0 = BA2(A, T_0)$ except with probability $2\delta_2$, where δ_2 is the success probability of $BA2$:

$$\frac{\|g[W_1]\|_1}{r(n)} \le y_1 \le r(n)\|g[W_1]\|_1 , \quad \frac{\|g[W_0]\|_1}{r(n)} \le y_0 \le r(n)\|g[W_0]\|_1.$$

Hence, except with probability $\frac{1}{4} + 2\delta_2$, we have the following constraints on y_0 and y_1:

$$y_0 \le r(n)Y \le r(n) \cdot \frac{3}{4} \cdot \|g[W]\|_1 \le 3r(n)X \le 3y_1 r^2(n) \le \tau(n, \epsilon) \cdot y_1, \text{ and}$$
$$y_1 \le r(n)X \le r(n) \cdot \frac{3}{4} \cdot \|g[W]\|_1 \le 3r(n)Y \le 3y_0 r^2(n) \le \tau(n, \epsilon) \cdot y_0,$$

in which case we set $b_\ell = 2$. If δ_2 is some small constant, say $\delta_2 \le \frac{1}{32}$, then for a fixed iteration ℓ, we set $b_\ell = 2$ except with probability $\frac{5}{16}$. Now, applying the Chernoff bound, we can show that the probability of having more than a $\frac{2}{5}$-fraction of iterations ℓ with $b_\ell \ne 2$ is at most an inverse polynomial. Hence, in this subcase the algorithm outputs $(-1, 0)$, except with inverse polynomial probability.

In the second subcase, we have $u_i > \frac{1}{16}\|g[W]\|_1$ and, for all $j \ne i$, $u_j \le \frac{\epsilon}{128}u_i$. Then, since u_i is not $\left(1 - \frac{\epsilon}{8}\right)$-heavy with respect to $g[W]$, we have:

$$u_j \le \frac{\epsilon}{128} \cdot u_i \le \frac{1}{16}(\|g[W]\|_1 - u_i).$$

Hence, we can apply Lemma 15 to the vector $U = (u_1, \ldots, u_{i-1}, 0, u_{i+1}, \ldots, u_n)$ (since $\|U\|_1 = \|g[W]\|_1 - u_i$, and moreover each entry in U is at most $\frac{1}{16}\|U\|_1$). Letting $X' = \sum_{j \ne i} h_j^\ell u_j$ and $Y' = \sum_{j \ne i} \bar{h}_j^\ell u_j$, we get that:

$$\Pr\left[\left(X' \le \frac{1}{4} \cdot \|U\|_1\right) \cup \left(Y' \le \frac{1}{4} \cdot \|U\|_1\right)\right] \le \frac{1}{4}.$$

This implies that $X \geq X' > \frac{1}{4}(\|g[W]\|_1 - u_i) \geq \frac{\epsilon}{32}\|g[W]\|_1$ and $Y \geq Y' > \frac{1}{4}(\|g[W]\|_1 - u_i) \geq \frac{\epsilon}{32}\|g[W]\|_1$. Moreover, except with probability $2\delta_2$, y_1 and y_0 are good approximations to $\|g[W_1]\|_1$ and $\|g[W_0]\|_1$, respectively. Thus, except with probability $\frac{1}{4} + 2\delta_2$, we have:

$$\begin{aligned} y_0 \leq r(n)Y \leq r(n)\left(1 - \frac{\epsilon}{32}\right)\|g[W]\|_1 &\leq r(n)\left(1 - \frac{\epsilon}{32}\right) \cdot \frac{32}{\epsilon} \cdot X \\ &\leq \frac{32r^2(n)}{\epsilon} \cdot y_1 \leq \tau(n,\epsilon) \cdot y_1, \text{ and} \\ y_1 \leq r(n)X \leq r(n)\left(1 - \frac{\epsilon}{32}\right)\|g[W]\|_1 &\leq r(n)\left(1 - \frac{\epsilon}{32}\right) \cdot \frac{32}{\epsilon} \cdot Y \\ &\leq \frac{32r^2(n)}{\epsilon} \cdot y_0 \leq \tau(n,\epsilon) \cdot y_0. \end{aligned}$$

This implies that, for a fixed iteration ℓ, the algorithm sets $b_\ell = 2$ except with probability $\frac{1}{4} + 2\delta_2$. Applying the Chernoff bound again, we see that the probability of having more than a $\frac{2}{5}$-fraction of iterations ℓ with $b_\ell \neq 2$ is at most an inverse polynomial. Thus, in this subcase, the algorithm outputs $(-1, 0)$ except with inverse polynomial probability.

We now consider the last subcase, where $u_i > \frac{1}{16}\|g[W]\|_1$ and there exists $j \neq i$ such that $u_j > \frac{\epsilon}{128}u_i$. Note that the probability that i and j get mapped to different matrices is given by $\Pr[H_\ell(i) \neq H_\ell(j)] = \frac{1}{2}$. Assume without loss of generality that $H_\ell(j) = 1$ (the case that $H_\ell(j) = 0$ is symmetric). In the event that i and j get mapped to different matrices and y_1, y_0 are good approximations to $\|g[W_1]\|_1, \|g[W_0]\|_1$ respectively, which happens with probability at least $\frac{1}{2} - 2\delta_2$, we have:

$$\begin{aligned} y_1 &\geq \frac{X}{r(n)} \geq \frac{u_j}{r(n)} \geq \frac{\epsilon}{128r(n)} \cdot u_i \geq \frac{\epsilon}{128r(n)} \cdot \frac{1}{16} \cdot \|g[W]\|_1 \\ &\geq \frac{\epsilon}{2048r(n)} \cdot Y \geq \frac{\epsilon}{2048r^2(n)} \cdot y_0 \implies y_0 \leq \frac{2048r^2(n)}{\epsilon} \cdot y_1 \leq \tau(n,\epsilon) \cdot y_1, \text{ and} \\ y_0 &\geq \frac{Y}{r(n)} \geq \frac{u_i}{r(n)} \geq \frac{\epsilon}{128r(n)} \cdot u_i \geq \frac{\epsilon}{128r(n)} \cdot \frac{1}{16} \cdot \|g[W]\|_1 \\ &\geq \frac{\epsilon}{2048r(n)} \cdot X \geq \frac{\epsilon}{2048r^2(n)} \cdot y_1 \implies y_1 \leq \frac{2048r^2(n)}{\epsilon} \cdot y_0 \leq \tau(n,\epsilon) \cdot y_0. \end{aligned}$$

Thus, except with probability at most $\frac{1}{2} + 2\delta_2$, the algorithm sets $b_\ell = 2$ for each iteration ℓ. We apply the Chernoff bound again to get that $b_\ell = 2$ for at least a $\frac{2}{5}$-fraction of iterations, except with inverse polynomial probability. Hence, the algorithm outputs $(-1, 0)$ except with inverse polynomial probability. ◀

▶ **Lemma 24.** *If the matrix $g[I_H A]$ does not have a key row but has an α-heavy row i_0, where $\alpha = 1 - \frac{\epsilon}{8}$, then Algorithm 1 either returns $(-1, 0)$ or returns a $(1 \pm \epsilon)$-approximation of $u_{I_H A, i_0}$ and the corresponding row i_0 with high probability.*

Proof. We know there is an α-heavy row, but not a key row. Note that there cannot be more than one α-heavy row for $\alpha = 1 - \frac{\epsilon}{8}$. If the algorithm returns $(-1, 0)$, then the lemma holds (note the algorithm is allowed to return $(-1, 0)$ since there is no key row). If the algorithm returns a pair of the form $(i, BA1(A, H))$, we know from Theorem 13 that the approximation of the weight of the α-heavy row is a $(1 \pm \epsilon)$-approximation of $\|g[W]\|_1$ as long as $BA1$ succeeds, which happens except with inverse polylogarithmic probability (the argument that the approximation is good follows similarly as in Lemma 22). We need only argue that we return the correct index, i_0. Again, the argument follows similarly as in Lemma 22. In particular, if $H_\ell(i) = b_\ell$ for a fixed iteration ℓ, then at least one of y_0, y_1 is a bad approximation or $H_\ell(i_0) = H_\ell(i)$, which happens with probability at most $2\delta_2 + \frac{1}{2}$ (where δ_2 is the failure probability of $BA2$). We then apply the Chernoff bound, similarly as before. ◀

With Lemmas 22, 23, and 24, we are done proving that Algorithm 1 fits the description of Definition 14, except with inverse polylogarithmic probability.

C Recursive Sketches

In this section, we give relevant notation and describe the Recursive Sum algorithm found in [11]. We first give some definitions which will be useful when describing the algorithm, the first of which will help us define a cover.

▶ **Definition 25.** Let Ω be a finite set of real numbers. For any positive integer t, we define $Pairs_t$ to be the set of all sets of pairs of the form:

$$\{(i_1, w_1), \dots, (i_t, w_t)\}, \quad \text{where } 1 \le i_1 < i_2 < \cdots < i_t \le n, i_j \in \mathbb{Z}, w_j \in \Omega.$$

We also further define

$$Pairs = \emptyset \cup \left(\bigcup_{t=1}^{n} Pairs_t \right).$$

We now provide the definition of a cover.

▶ **Definition 26.** We say a non-empty set $Q \in Pairs_t$ for some $t \in [n]$ (i.e., $Q = \{(i_1, w_1), \dots, (i_t, w_t)\}$) is an (α, ϵ)-cover with respect to the vector $V = (v_1, \dots, v_n)$ (where each $v_i \ge 0$) if the following is true:

1. $\forall j \in [t]$: $(1-\epsilon)v_{i_j} \le w_j \le (1+\epsilon)v_{i_j}$.
2. $\forall i \in [n]$: If v_i is α-heavy then $\exists j \in [t]$ such that $i_j = i$ (here, α-heavy means $v_i \ge \alpha \sum_j v_j$).

We also define the following index set, and some other notation that is useful for the algorithm.

▶ **Definition 27.** For a non-empty set $Q \in Pairs$, we define $Ind(Q)$ to be the set of indices of Q. More formally, for $Q \in Pairs$, we let $Ind(Q) = \{i : \exists j \le t$ such that, for the j^{th} pair (i_j, w_j) of Q, we have $i_j = i\}$. For $i \in Ind(Q)$, we denote by $w_Q(i)$ the corresponding approximation. More formally, if $i = i_j$, then $w_Q(i) = w_j$. Note that, since $i_j < i_{j+1}$, this is a valid definition. For completeness, we let $w_Q(i) = 0$ for $i \notin Ind(Q)$ and $Ind(\emptyset) = \emptyset$.

▶ **Definition 28.** We say $H : [n] \to \{0, 1\}$ is a pairwise independent zero-one vector if the zero-one entries are uniformly distributed and pairwise independent. In particular, we have $\Pr[H(i) = 0] = \Pr[H(i) = 1] = \frac{1}{2}$, and moreover the entries are pairwise independent.

Using notation from [11], for a vector $V = (v_1, \dots, v_n)$, we let $|V|$ denote the L_1 norm of V, $|V| = \sum_{i=1}^{n} v_i$. Note that the product of two pairwise independent zero-one vectors H_1 and H_2 is simply given by the Hadamard product. Moreover, we let F_0 denote the 0^{th} frequency moment, so that $F_0(V)$ counts the number of distinct elements in the vector V. In the following algorithm, we let $HH(D, \alpha, \epsilon, \delta)$ be an algorithm that produces an (α, ϵ)-cover with respect to the vector $V = V(D)$ with probability at least $1 - \delta$ (where $V(D)$ is a vector of dimensionality n defined by the stream D). For some integer parameter ϕ, let $H_1, \dots, H_\phi$ be i.i.d. random vectors with zero-one entries that are uniformly distributed and pairwise independent. We define vectors V_j for $0 \le j \le \phi$ via the following iterative process: $V_0 = V$,

Algorithm 3 Recursive Sum (D, ϵ)

```
1. Generate $\phi = O(\log(n))$ pairwise independent zero-one vectors $H_1, \dots, H_\phi$. Denote by
   $D_j$ the stream $D_{H_1 H_2 \cdots H_\phi}$
2. Compute, in parallel, $Q_j = HH(D_j, \frac{\epsilon^2}{\phi^3}, \epsilon, \frac{1}{\phi})$
3. If $F_0(V_\phi) > 10^{10}$ then output 0 and stop. Otherwise, compute precisely $Y_\phi = |V_\phi|$
4. For each $j = \phi - 1, \dots, 0$, compute
       $Y_j = 2Y_{j+1} - \sum_{i \in Ind(Q_j)} (1 - 2h_i^j) w_{Q_j}(i)$
5. Output $Y_0$
```

and $V_j = HAD(V_{j-1}, H_j)$ for $j = 1, \dots, \phi$. We denote by h_i^j the i^{th} entry of H_j. For a function $H : [n] \to \{0, 1\}$, define D_H to be a substream of the stream D that contains only elements $i \in D$ such that $H(i) = 1$.

We are now ready to define the Recursive Sum algorithm (Algorithm 6 from [11]). Theorem 4.1 from [11]:

▶ **Theorem 29.** *Algorithm 3 computes a $(1 \pm \epsilon)$-approximation of $|V|$ and errs with probability at most 0.3. The algorithm uses $O(\log(n)\mu(n, \frac{\epsilon^2}{\log^3(n)}, \epsilon, \frac{1}{\log(n)}))$ bits of memory, where μ is the space required by the above algorithm HH.*

Local Convergence and Stability of Tight Bridge-Addable Graph Classes

Guillaume Chapuy[*1] **and Guillem Perarnau**[2]

1 **IRIF, UMR CNRS 8243, Université Paris-Diderot, France; and CRM, UMI CNRS 3457, Université de Montréal, Canada**
guillaume.chapuy@liafa.univ-paris-diderot.fr

2 **School of Mathematics, University of Birmingham, Birmingham, U.K.**
g.perarnau@bham.ac.uk

Abstract

A class of graphs is *bridge-addable* if given a graph G in the class, any graph obtained by adding an edge between two connected components of G is also in the class. The authors recently proved a conjecture of McDiarmid, Steger, and Welsh stating that if $\mathcal{G}$ is bridge-addable and G_n is a uniform n-vertex graph from $\mathcal{G}$, then G_n is connected with probability at least $(1+o(1))e^{-1/2}$. The constant $e^{-1/2}$ is best possible since it is reached for the class of forests.

In this paper we prove a form of uniqueness in this statement: if $\mathcal{G}$ is a bridge-addable class and the random graph G_n is connected with probability close to $e^{-1/2}$, then G_n is asymptotically close to a uniform forest in some "local" sense. For example, if the probability converges to $e^{-1/2}$, then G_n converges for the Benjamini-Schramm topology, to the uniform infinite random forest F_∞. This result is reminiscent of so-called "stability results" in extremal graph theory, with the difference that here the "stable" extremum is not a graph but a graph class.

1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases bridge-addable classes, random graphs, stability, local convergence, random forests.

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.26

1 Introduction and Main Results

In this paper graphs are simple. A graph is labeled if its vertex set is of the form $[1..n]$ for some $n \geq 1$. An unlabeled graph is an equivalence class of labeled graphs by relabeling. Unless mentioned otherwise, graphs are labeled. A class of (labeled) graphs $\mathcal{G}$ is *bridge-addable* if given a graph G in the class, and an edge e of G whose endpoints belong to two distinct connected components, then $G \cup \{e\}$ is also in the class. McDiarmid, Steger and Welsh [11] conjectured that every bridge-addable class contains at least a proportion $(1+o(1))e^{-1/2}$ of connected graphs. This has recently been proved by the authors. In the next statement and later, we denote by $\mathcal{G}_n$ the set of graphs in $\mathcal{G}$ with n vertices, and G_n a uniform random element of $\mathcal{G}_n$.

▶ **Theorem 1** (Chapuy, Perarnau [4]). *For every $\epsilon > 0$, there exists an n_0 such that for every bridge-addable class $\mathcal{G}$ and every $n \geq n_0$, we have*

$$\mathbf{Pr}\,(G_n \textit{ is connected}) \geq (1-\epsilon)e^{-1/2}. \tag{1}$$

[*] G.C. acknowledges support from grant ANR-12-JS02-001-01 "Cartaplus" and from the City of Paris, program "Émergences 2013".

If $\mathcal{G}$ is the class of all forests (which is bridge-addable), then Theorem 1 is asymptotically tight, since it is shown in [12] that if F_n is a uniform random forest on n vertices, then as n tends to infinity:

$$\mathbf{Pr}\,(F_n \text{ is connected}) \longrightarrow e^{-1/2}. \tag{2}$$

The aim of this paper is to show that, in some appropriate sense, this class is the only one for which Theorem 1 is asymptotically tight. More precisely, we will show that any addable class of graphs that comes close to achieving the constant $e^{-1/2}$ is "close" to a uniform random forest in some local sense.

▶ **Definition 2.** For any $\zeta > 0$, we say that a bridge-addable class of graphs $\mathcal{G}$ is *ζ-tight with respect to connectivity* (or simply *ζ-tight*) if there exists an n_0 such that for every $n \geq n_0$ we have

$$\mathbf{Pr}\,(G_n \text{ is connected}) \leq (1+\zeta)e^{-1/2}\ ,$$

where we recall that G_n is chosen uniformly at random from $\mathcal{G}_n$.

In order to state our results, we first need to introduce some notation and terminology. If H is a graph we let $|H|$ be its number of vertices. We denote by $\mathcal{U}$ the set of *unlabeled, unrooted* trees and by $\mathcal{T}$ the set of *unlabeled, rooted* trees, *i.e.* trees with a marked vertex called the root. For every tree $U \in \mathcal{U}$, we denote by $\mathrm{Aut}_u(U)$ the number of automorphisms of U, and for every rooted tree $T \in \mathcal{T}$, we denote by $\mathrm{Aut}_r(T)$ the number of automorphisms of T that fix its root. Moreover given k trees $U_1,\dots,U_k$ in $\mathcal{U}$, we denote by $\mathrm{Aut}_u(U_1,\dots,U_k)$ the number of automorphisms of the forest formed by disjoint copies of $U_1,\dots,U_k$.

Given a graph H, we let $\mathit{Small}(H)$ denote the graph formed by all the components of H that are not the largest one (in case of a tie, we say that the largest component of the graph is the one with the largest vertex label among all candidates). In what follows, we will always see $\mathit{Small}(H)$ as an unlabeled graph. Given a graph G and a rooted tree $T \in \mathcal{T}$, we let $\alpha^G(T)$ be the number of pendant copies of the tree T in G. More precisely, $\alpha^G(T)$ is the number of vertices v of G having the following property: there is at least one cut-edge e incident to v, and if we remove the such cut-edge that separates v from the largest possible component, the vertex v lies in a component of the graph that is a tree, rooted at v, which is isomorphic to T. The following is classical:

▶ **Theorem 3** (see [5]). *Let F_n be a uniform random forest with n vertices. Then for any fixed unlabeled unrooted forest $\mathbf{f}$ we have as n goes to infinity:*

$$\mathbf{Pr}\big(\mathit{Small}(F_n) \equiv \mathbf{f}\big) \longrightarrow p_\infty(\mathbf{f}) := e^{-1/2}\frac{e^{-|\mathbf{f}|}}{\mathit{Aut}_u(\mathbf{f})}, \tag{3}$$

where $\equiv$ denotes unlabeled graph isomorphism. Moreover, p_∞ is a probability distribution on the set of unlabeled unrooted forests.

For any fixed rooted tree $T \in \mathcal{T}$ we have as n goes to infinity:

$$\frac{\alpha^{F_n}(T)}{n} \xrightarrow{(p)} a_\infty(T) := \frac{e^{-|T|}}{\mathit{Aut}_r(T)}, \tag{4}$$

where (p) indicates convergence in probability.

Our main result says that, for bridge-addable classes, if we have an approximate version of (2), then we also have an approximate version of (3) and (4). In the next statement and everywhere in the paper, the constants $\epsilon, \eta, \rho, \nu, \zeta$ are implicitly assumed (in addition to other written quantifications or assumptions) to be positive and smaller than c where c is a small, absolute, constant.

▶ **Theorem 4** (Main result). *For every $\epsilon, \eta > 0$, there exists a $\zeta > 0$ and an n_0 such that for every ζ-tight bridge-addable class $\mathcal{G}$ and $n \geq n_0$, the following holds:*

(i) *The small components of G_n are close to those of a large random forest, in the sense that for every unrooted unlabeled forest $\mathbf{f}$ we have:*

$$\Big|\mathbf{Pr}\big(\, Small(G_n) \equiv \mathbf{f}\big) - p_\infty(\mathbf{f})\Big| < \epsilon.$$

(ii) *The statistics of pendant trees in G_n are close to those of a large random forest, in the sense that:*

$$\mathbf{Pr}\left(\forall T \in \mathcal{T} : \left|\frac{\alpha^{G_n}(T)}{n} - a_\infty(T)\right| < \eta\right) > 1 - \epsilon.$$

▶ Remark. It is easy to see (up to adapting the dependence of ζ in ϵ, η) that we can replace *i*) by:

(i') The total variation distance between the law of $Small(G_n)$ and the probability law p_∞ is at most ϵ.

Similarly we could replace *ii*) by:

(ii') The total variation distance between the measure $\alpha^{G_n}(\cdot)/n$ and the probability law $a_\infty(\cdot)$ (both are measures on $\mathcal{T}$) is at most η with probability at least $1 - \epsilon$.

▶ Remark. Our main result, Theorem 4, can be viewed both as a *unicity result* (since it states that in the limit, and through the lens of local observables, the class of forests is the only one to reach the optimum value $e^{-1/2}$) and as a *stability result* (since it also states that the only classes than come *close* to the extremal value $e^{-1/2}$ are *close* to forests, again through local observables of random graphs). Here we use the terminology "stability result" on purpose, by analogy with the field of extremal graph theory. Indeed the study of graphs that come *close* to achieving extremal properties is a classical topic in this field. *Stability results*, pioneered in the papers [8, 7, 6, 13], show that in many cases, the graphs that are close to being extremal have a structure close to the actual extremal graphs, in some quantifiable sense. Our main result suggests that the question of stability of extremal graph classes, with respect to appropriate graph limit topologies (here, local convergence), should be further examined.

Before going into the proof of the theorem, let us look at some closely related statements and corollaries. Call a bridge-addable class $\mathcal{G}$ *tight* is it is ζ-tight for any ζ, that is to say:

$$\mathbf{Pr}(G_n \text{ is connected}) \to e^{-1/2}.$$

Then we have the following consequence of Theorem 4. Note that it is weaker (it is a unicity, but not a stability result).

▶ **Theorem 5** (Convergence of local statistics in tight graph classes). *Let $\mathcal{G}$ be a tight bridge-addable class of graphs. Then, when n goes to infinity, $Small(G_n)$ converges in total variation distance to the probability law $p_\infty(.)$ given by (3), that is:*

$$d_{TV}\big(\, Small(G_n), p_\infty\big) \longrightarrow 0.$$

Moreover, for any rooted tree $T \in \mathcal{T}$, the proportion $\frac{\alpha^{G_n}(T)}{n}$ of local pendant copies of the tree T converges in probability to the deterministic constant $a_\infty(T)$ given by (4):

$$\frac{\alpha^{G_n}(T)}{n} \xrightarrow{(p)} a_\infty(T).$$

Theorem 5 states that, from the point of view of statistics of pendant trees and of non-largest components, tight classes are indistinguishable from random forests in the limit. Let us develop in this direction. Let V_n be a uniform random vertex in G_n. Then for a given $T \in \mathcal{T}$, conditionally to G_n, the quantity $\alpha^{G_n}(T)/n$ is the probability that from V_n hangs a copy of the tree T. Readers familiar with the Benjamini-Schramm (BS) convergence of graphs [3] will note the similarity with this notion. If (G, x) and (H, y) are two rooted graphs, define the BS-distance $d_{BS}((G, x); (H, y))$ as 2^{-K} where K is the largest integer such that the balls of radius K in (G, x) and (H, y) are isomorphic (as rooted graphs). This distance (also called the *ball distance*, see [10]) defines a topology (in fact, a Polish space) on the set of rooted graphs, and enables us to talk about convergence in distribution of random rooted graphs, in the BS-sense. An equivalent definition of this convergence is the following: a sequence of random rooted graphs (H_n, x_n) converges to (H_∞, x_∞) if and only if for any rooted graph (H, x) of radius r, the probability that the ball of radius r in (H_n, x_n) is isomorphic to (H, x) converges to the probability of the same event in (H_∞, x_∞).

It is easy to see (for example using generating functions, see [5]) that if F_n is a uniform random forest on n vertices rooted at a random uniform vertex V_n, then

$$(F_n, V_n) \to (F_\infty, V_\infty)$$

in distribution in the BS-sense, where (F_∞, V_∞) is the "infinite uniform random forest". Namely, (F_∞, V_∞) can be constructed as follows: consider a semi-infinite path, starting at a vertex V_∞, and identify each vertex of this path with the root of an independent Galton-Watson tree of offspring distribution $Poisson(1)$.

In our context, passing from pendant trees to balls is an easy task, and one can deduce the following from Theorem 5.

▶ **Corollary 6** (Local convergence of tight graph classes). *Let $\mathcal{G}$ be a tight bridge-addable graph class. Let G_n be a uniform random graph from $\mathcal{G}_n$ and let V_n be a uniform random vertex of G_n. Then (G_n, V_n) converges to (F_∞, V_∞) in distribution in the Benjamini-Schramm sense.*

The purpose of stating Corollary 6 is to illustrate the link between our local observables and the BS topology, but we could have stated stronger intermediate results. For example Corollary 6 uses only the second part of Theorem 5, and says nothing about small connected components. In fact, it is true that for tight classes, the pair $((G_n, V_n), Small(G_n))$ converges in distribution to $(F_\infty, V_\infty) \otimes p_\infty$ for the product of the BS and the total variation topologies. This follows easily from our proofs.

Also note the last corollary (and the other statements of the same kind that have just mentioned) is of a much weaker nature than Theorem 5. Indeed, Theorem 5 asserts that with high probability, conditional on the random graph G_n, the graph G_n is similar to a random forest when viewed from a random vertex, whereas Corollary 6 is an unconditioned statement that averages both over the graph G_n and the vertex V_n. It is possible to formulate a version Theorem 5 in terms of the BS convergence as follows. Let μ_{G_n} be the law, conditional on G_n, of the random rooted graph (G_n, V_n) where V_n is a uniform vertex of G_n (then μ_{G_n} is a random probability measure on the set of rooted graphs). Then it follows easily from Theorem 5 that if $\mathcal{G}$ is a tight bridge-addable and n is large enough, μ_{G_n} converges in probability to the *deterministic* probability measure μ_∞, defined as the law of (F_∞, V_∞). The underlying distance for the convergence in probability is the Skorokhod distance induced by the BS distance on the set of probability measures on rooted graphs. We will not give more details on these questions, since the related considerations of convergence of probability measures would lead us too far from our main prospect.

▶ Remark. Our main theorem asserts that tight bridge addable classes are "locally similar" to random forests in some precise sense. However, they can be very different from some other perspective. For example, consider the set $\tilde{\mathcal{F}}_n$ of graphs on $[1..n]$ defined as follows: $\tilde{\mathcal{F}}_n$ contains the graph in which all edges linking vertices in $[1..\lceil n^{2/3}\rceil]$ are present and all other vertices are isolated, and $\tilde{\mathcal{F}}_n$ is the smallest bridge-addable class containing this graph. In other words, $\tilde{\mathcal{F}}_n$ is the set of graphs inducing a clique on $[1..\lceil n^{2/3}\rceil]$, and such that contracting this clique gives a forest. Then $\tilde{\mathcal{F}} = \cup_{n\geq 1}\tilde{\mathcal{F}}_n$ is a bridge-addable class, and it is easy to see that it is tight, so our main results apply. However one can argue that the random graph $\tilde{F}_n$ in $\tilde{\mathcal{F}}_n$ is *very* different from a random forest in several senses: first, it has $\Theta(n^{4/3})$ edges whereas a forest has linearly many. Second, with probability $1 - O(n^{-1/3})$ an edge taken uniformly at random from $\tilde{F}_n$ belongs to a clique of size $\lceil n^{2/3}\rceil$, which is very different from what happens in a forest. This last point does not contradict our results, but only recalls that it is important here to think of locality as a measure of what happens around "typical vertices" and not "typical edges".

▶ Remark. One can modify the example of the previous remark by replacing the clique of size $\lceil n^{2/3}\rceil$ by a path of length $\lceil n^{2/3}\rceil$. One thus obtains a tight bridge-addable class of graphs, in which the *diameter* of the largest component is of order $\Theta(n^{2/3})$, which is again very different from a random forest in which the giant tree has diameter $\Theta(\sqrt{n})$ with high probability. In both examples, the function $n^{2/3}$ plays no special role and may be replaced by $n^{1-\epsilon}$ for any $\epsilon > 0$.

We conclude this list of results with a simpler statement, that does not require the full strength of our main theorems (it is a relatively easy consequence of the results of [4]).

▶ **Theorem 7.** *Let $\mathcal{G}$ be a tight bridge-addable class and G_n a uniform random graph from $\mathcal{G}_n$. Then for any $k \geq 0$, we have*

$$\mathbf{Pr}\left(G_n \text{ has } k+1 \text{ connected components}\right) \longrightarrow e^{-1/2}\,\frac{2^{-k}}{k!}.$$

In other words, the number of connected components of G_n converges in distribution to Poisson$(1/2)$.

Structure of this abstract. In this abstract, we will present the main steps of the proof of Theorem 4, refereeing the reader to the full version [5] for complete proofs, including several easy results on enumeration of forests and on random forests.

2 Theorem 4 for bridge-addable classes of forests

2.1 Number of components in bridge-addable graph classes

We first introduce some notation, following [4]. For a bridge-addable class of graphs $\mathcal{G}$ and for $i \geq 1$, we note $\mathcal{G}_n^{(i)}$ the set of n-vertex graphs in $\mathcal{G}$ having i connected components. An elegant double-counting argument going back to [11] asserts that for all $i \geq 1$, and $n \geq 1$ we have:

$$i \cdot \left|\mathcal{G}_n^{(i+1)}\right| \leq \left|\mathcal{G}_n^{(i)}\right|. \tag{5}$$

This statement follows by double-counting the edges of an auxiliary bipartite graph on the vertex set $\mathcal{G}_n^{(i)} \uplus \mathcal{G}_n^{(i+1)}$, where two graphs G, H are linked by an edge if and only if one can be obtained from the other by adding a bridge: on the one hand, an element of $\mathcal{G}_n^{(i+1)}$ has degree at least $i(n-i)$ in this auxiliary graph, since $\mathcal{G}$ is bridge-addable; on the other hand, an element of $\mathcal{G}_n^{(i)}$ has degree at most $(n-i)$ (which is the maximum number of cut-edges in a graph with i connected components and n vertices). Thus (5) follows. The main achievement of the paper [4] was to improve this bound by roughly a factor $\frac{1}{2}$, asymptotically.

▶ **Lemma 8** (Proposition 4 in [4]). *For every η and every k there exists an n_0 such that for every bridge-addable class $\mathcal{G}$, every $n \geq n_0$ and every $i \leq k$, one has:*

$$i|\mathcal{G}_n^{(i+1)}| \leq \left(\frac{1}{2}+\eta\right)|\mathcal{G}_n^{(i)}| \,. \tag{6}$$

The following lemma, which follows relatively easily from Theorem 1, provides a converse inequality to (6) for ζ-tight classes. Note that it implies Theorem 7.

▶ **Lemma 9.** *For every η and every k there exists a ζ and an n_0 such that for every ζ-tight bridge-addable class $\mathcal{G}$, every $n \geq n_0$ and every $i \leq k$, one has*

$$\left(\frac{1}{2}-\eta\right)|\mathcal{G}_n^{(i)}| \leq i|\mathcal{G}_n^{(i+1)}| \leq \left(\frac{1}{2}+\eta\right)|\mathcal{G}_n^{(i)}| \,.$$

2.2 Partitioning the graph class into highly structured subclasses

Balister, Bollobás and Gerke [2, Lemma 2.1] proposed an elegant construction that reduces the statement of Theorem 1 to the case where all graphs in $\mathcal{G}$ are forests. As we will see in the next section, their idea can be adapted to the present context. We will therefore start by proving Theorem 4 for classes $\mathcal{G}$ composed by forests:

Throughout the rest of Section 2 we will assume that all graphs in $\mathcal{G}$ are forests.

We will first focus on the graphs in $\mathcal{G}_n$ that have either one or two connected components, and, in view of this, we use the shorter notation $\mathcal{A}_n := \mathcal{G}_n^{(1)}$ and $\mathcal{B}_n := \mathcal{G}_n^{(2)}$. We now introduce a partitioning of $\mathcal{A}_n$ and $\mathcal{B}_n$ in terms of some local statistics, which requires the following set-up, that is modeled on [4, proof of Prop 3]. Here ϵ and k_* are two constants, whose value may vary along the course of the paper, that will *in fine* be chosen very small and very large, respectively:

- $\mathcal{U}_\epsilon$ is the set of unrooted trees of order at most $\lceil \epsilon^{-1} \rceil$: $\mathcal{U}_\epsilon := \{U \in \mathcal{U},\ |U| \leq \lceil \epsilon^{-1} \rceil\}$.
- $\mathcal{T}_*$ is the set of rooted trees of order at most k_*: $\mathcal{T}_* := \{T \in \mathcal{T},\ |T| \leq k_*\}$.

More generally for any given $\ell \geq 1$ we will use the notation $\mathcal{T}_{\leq \ell}$,$\mathcal{U}_{\leq \ell}$ to denote the set of rooted (resp., unrooted) trees of order at most ℓ, so that $\mathcal{U}_\epsilon = \mathcal{U}_{\leq \lceil \epsilon^{-1} \rceil}$ and $\mathcal{T}_* = \mathcal{T}_{\leq k_*}$.

Roughly speaking, we will use elements of $\mathcal{U}_\epsilon$ and $\mathcal{T}_*$ as "test graphs" to measure the shape of small components of G_n and the number of pending subtrees of G_n, respectively. For $\ell \geq 1$ we write $\mathcal{E}_\ell = \{0, \dots, n-1\}^{\mathcal{T}_{\leq \ell}}$, and we will be particularly concerned with the set $\mathcal{E}_* := \mathcal{E}_{k_*}$, namely the set of integer vectors with one coordinate for each "test tree" in $\mathcal{T}_*$. For $\alpha \in \mathcal{E}_*$ and $w \geq 1$ (width) we define the *box* $[\alpha]^w \subset \mathcal{E}_*$ and its *q-neighborhood* $[\alpha]_q^w$ as the parallelepipeds:

$$[\alpha]^w := \{\alpha' \in \mathcal{E}_* :\ \forall T \in \mathcal{T}_*,\ \alpha(T) \leq \alpha'(T) < \alpha(T) + w\} \,,$$
$$[\alpha]_q^w := \{\alpha' \in \mathcal{E}_* :\ \forall T \in \mathcal{T}_*,\ \alpha(T) - q \leq \alpha'(T) < \alpha(T) + w + q\} \,.$$

Finally, if $\mathcal{S}_n$ is a set of graphs (where the letter $\mathcal{S}$ could be $\mathcal{A}$, $\mathcal{B}$, and also carry other decorations), we let $\mathcal{S}_{n,[\alpha]^w}$ be the set of graphs G in $\mathcal{S}$ such that $\alpha^G(T) \in [\alpha]^w$ for all $T \in \mathcal{T}_*$, and we use the same notation with $[\alpha]_q^w$. Also, for every forest $\{U_1, \dots, U_k\}$, we denote by ${\mathcal{S}_n}^{\{U_1,\dots,U_k\}}$ the set of graphs G in $\mathcal{S}_n$ such that $\mathit{Small}(G)$ is isomorphic to $\{U_1, \dots U_k\}$. In the case of graphs with two connected components, we just use the notation $\mathcal{S}_n^U$ for $\mathcal{S}_n^{\{U\}}$, where $U \in \mathcal{U}$.

2.3 Good and bad boxes

The main concern of the paper [4] was to obtain a version of the double-counting argument of Section 2.1 that is *local* in the sense that it relates cardinalities of graphs corresponding to fixed boxes. Given ϵ (hence $\mathcal{U}_\epsilon$) and $\mathcal{T}_*$, [4, Lemma 16] asserts that there exist integers K, w, n_0 (independent of $\mathcal{G}$) and a set of K disjoint boxes of width w in $\mathcal{E}_*$, noted $\{[\beta_i]^w,\ 1 \le i \le K\}$, such that for $n \ge n_0$ and $U \in \mathcal{U}_\epsilon$ we have:

$$\sum_{i=1}^{K} |\mathcal{B}^U_{n,[\beta_i]^w}| \ge (1-\epsilon)|\mathcal{B}^U_n|, \tag{7}$$

and such that for each $1 \le i \neq j \le K$, $[\beta_i]^w_q \cap [\beta_j]^w_q = \emptyset$, where $q = q_\epsilon := \lceil \epsilon^{-1} \rceil$. In other words, these boxes are $2q$-apart from each other, and yet capture a proportion at least $(1-\epsilon)$ of the set $\mathcal{B}^U_n$ for each $U \in \mathcal{U}_\epsilon$. We will also use the fact (implicit in [4]) that the $[\beta_i]^w_q$ partition the set $\mathcal{E}_*$.

In the present paper, one of the main tasks consists in showing that the global estimates obtained in [4] can be "lowered" down to boxes for ζ-tight classes. For $\gamma, \epsilon > 0$, we say that a box $[\alpha]^w$ is (γ,ϵ)-*good* (or simply *good*) if the two following conditions hold:

(i) $|\mathcal{B}_{n,[\alpha]^w}| \ge \left(\frac{1}{2} - \gamma\right) \cdot |\mathcal{A}_{n,[\alpha]^w_q}|$

(ii) $\sum_{U \notin \mathcal{U}_\epsilon} |\mathcal{B}^U_{n,[\alpha]^w}| < \gamma |\mathcal{B}_{n,[\alpha]^w}|$.

Note that Property i) is a local version of the first inequality of Lemma 9, while Property ii) ensures that the number of graphs in sets that we do not control, is small (as it happens in a global scale). Hence good boxes are, in some sense, boxes that realize the tightness property *locally.* We will be interested in the boxes among the $[\beta_i]$ that are (γ,ϵ)-good:

$$\mathrm{Good}_{\gamma,\epsilon} := \{i \in [1..K] : [\beta_i] \text{ is } (\gamma,\epsilon)\text{-good}\} .$$

An important step in the proof of Theorem 4 is the following result:

▶ **Lemma 10.** *For every γ and every η, if $\epsilon < \epsilon_0(\gamma,\eta)$ and if $k_* \ge k_0(\epsilon)$, then there exist ζ and an n_0 such that for every ζ-tight bridge-addable class $\mathcal{G}$ and every $n \ge n_0$, one has*

$$\frac{\sum_{i \notin Good_{\gamma,\epsilon}} |\mathcal{A}_{n,[\beta_i]^w_q}|}{|\mathcal{A}_n|} < \eta\ ,\quad \frac{\sum_{i \notin Good_{\gamma,\epsilon}} |\mathcal{B}_{n,[\beta_i]^w}|}{|\mathcal{B}_n|} < \eta\ .$$

From this lemma, one deduces, after a lengthy and technical proof, the following result (which is a first version of Theorem 4 for subclasses of forests and for $\mathbf{f}$ being a tree). We define the set of vectors in $\mathcal{E}_\ell$ that are δ-close from the distribution p_∞ (recall that for $T \in \mathcal{T}$, $p_\infty(T) = \frac{e^{-|T|}}{\mathrm{Aut}_r(T)}$),

$$\Xi(\delta,\ell) = \left\{ \beta \in \mathcal{E}_\ell : \left| \frac{\beta(T)}{n} - p_\infty(T) \right| < \delta,\ \text{for every } T \in \mathcal{T}_{\le \ell} \right\} .$$

▶ **Proposition 11.** *For every θ_1 and every $U \in \mathcal{U}$, there exist a $\zeta > 0$ and an n_0 such that for every ζ-tight class $\mathcal{G}$ of forests and every $n \ge n_0$, one has*

$$\left| \frac{|\mathcal{B}^U_n|}{|\mathcal{G}_n|} - e^{-1/2} \frac{e^{-|U|}}{Aut_u(U)} \right| < \theta_1\ .$$

Moreover, for every θ_1, every δ, every ℓ and every $U \in \mathcal{U}$, there exist a $\zeta > 0$ and an n_0 such that for every ζ-tight class $\mathcal{G}$ of forests and every $n \ge n_0$, one has

$$\left| \frac{\sum_{\beta \in \Xi(\delta,\ell)} \left|\mathcal{B}^U_{n,\beta}\right|}{|\mathcal{B}^U_n|} - 1 \right| < \theta_1\ .$$

Main ideas of the proof. The proof itself is long (see [5]). The main idea is to go back to the optimization problem of [4] and show that, in order for the class to be ζ-tight, one has to be close to the extremal point of that problem. Roughly speaking, [4] provides some inequalities between the local ratios $|\mathcal{B}_{n,[\alpha]^w}|/|\mathcal{A}_{n,[\alpha]_q^w}|$, where $[\alpha]^w$ is a box, in terms of an optimization problem for the quantities $|\mathcal{B}^U_{n,[\alpha]^w}|/|\mathcal{A}_{n,[\alpha]_q^w}|$ for $U \in \mathcal{U}_\epsilon$. By the preceding lemma, if a class is ζ-tight for ζ small enough, we can almost cover the space $\mathcal{E}_*$ with boxes that capture most of the mass of the sets $\mathcal{A}_n$ and $\mathcal{B}_n$, and such that each box is good. By Property i) of good boxes, a good box is close to reaching the ratio $e^{-1/2}$ which is the optimum in the optimization problem of [4]. The main task of the proof is then to go back to the optimization problem of [4] and quantify the stability of its extrema. After a tedious technical work, one finds that, provided ϵ, k_* are respectively small and large enough, the optimization problem is sufficiently stable to conclude that most of the mass in the sets $\mathcal{B}_n^U$ is concentrated around the unique extreme of the optimization problem. More precisely, one finds that the set $\mathcal{B}_n^U$ has most of its mass in subsets $\mathcal{B}^U_{n,[\alpha]^w}$ such that $\alpha(T)$ is close to $a_\infty(T)$ for each $T \in \mathcal{T}_*$, and that for such subsets the ratios $|\mathcal{B}^U_{n,[\alpha]^w}|/|\mathcal{A}_{n,[\alpha]_q^w}|$ are close to $e^{-1/2}\frac{e^{-|U|}}{\mathrm{Aut}_u(U)}$. The result can then be extended to the ratios $|\mathcal{B}_n^U|/|\mathcal{A}_n|$ by an averaging argument, and to the ratios $|\mathcal{B}_n^U|/|\mathcal{G}_n|$ since for ζ-tight classes $|\mathcal{A}_n|/|\mathcal{G}_n|$ is close to $e^{-1/2}$.

We refer the reader again to the full version of the article [5] for the many subtleties hidden in this seemingly simple proof by tightness arguments. ◀

The proof of the previous proposition, although it involves a lot of work, is the part of the present paper that is conceptually more relying on [4]. The next result, that is equivalent to our main theorem (for classes of forests) relies on arguments of a different nature:

▶ **Theorem 12.** *For every $k \geq 1$, every θ_k and every $U_1, \ldots, U_k \in \mathcal{U}$, there exist a $\zeta > 0$ and an n_0 such that for every ζ-tight class $\mathcal{G}$ of forests and every $n \geq n_0$, one has*

$$\left| \frac{\left|\mathcal{G}_n^{k+1,\{U_1,\ldots,U_k\}}\right|}{|\mathcal{G}_n|} - e^{-1/2}\frac{e^{-\sum_{i=1}^k |U_i|}}{Aut_u(U_1,\ldots,U_k)} \right| < \theta_k \,. \tag{8}$$

Moreover, for every $k, \ell \geq 1$, every θ_k, δ and every $U_1, \ldots, U_k \in \mathcal{U}$, there exist a $\zeta > 0$ and an n_0 such that for every ζ-tight class $\mathcal{G}$ of forests and every $n \geq n_0$, one has

$$\left| \frac{\sum_{\beta \in \Xi(\delta,\ell)} \left|\mathcal{G}_{n,\beta}^{k+1,\{U_1,\ldots,U_k\}}\right|}{\left|\mathcal{G}_n^{k+1,\{U_1,\ldots,U_k\}}\right|} - 1 \right| < \theta_k \,. \tag{9}$$

Main ideas of the proof. The proof uses induction on k, with the base case given by Proposition 11. The main idea is that if $\mathcal{G}$ is bridge-addable and $k \geq 2$, and if $\{U_1, \ldots, U_{k-1}\}$ is a forest composed by k trees on a subset W of $[1..n]$, we can form a bridge-addable class by looking at all graphs G in $\mathcal{G}_n$ such that W induces a union of connected components of G, given by $\{U_1, \ldots, U_{k-1}\}$. Roughly speaking, connected graphs in this new class correspond to graphs in $\mathcal{G}_n^{(k)}$ while graphs with two connected components correspond to graphs in $\mathcal{G}_n^{(k+1)}$. Therefore, by applying Proposition 11 to this class, we may obtain information on the ratios of cardinalities of these sets. Moreover, the induction hypothesis ensures that we have a very precise structural information on the typical graphs in $\mathcal{G}_n^{(k)}$. The full proof is given in [5]. ◀

The last theorem implies Theorem 4 for bridge-addable classes of forests. Indeed, the first part of it implies *i*): by selecting ℓ large enough, we use (8) to control $\left|\mathcal{G}_n^{k+1,\{U_1,\ldots,U_k\}}\right|$

for all $k \leq \ell$ and $U_1, \dots, U_k \in \mathcal{T}_{\leq \ell}$, and since ℓ is large, the set $\mathcal{G}_n^{k+1,\{U_1,\dots,U_k\}}$ is of negligible size for the rest for forest with more than ℓ vertices or including some tree with order larger than ℓ. In a similar way, we can use (9) to prove the statement *ii*).

3 From classes of forests to classes of graphs

In this section we extend the results of the previous section (where we obtained Theorem 4 for classes of forests) to general bridge-addable classes, concluding the proof of Theorem 4.

The method of proof of Theorem 12 will allow us to derive a statement about *removable edges* which will be crucial to transfer the result from forest to general classes. We say that an edge in a graph $G \in \mathcal{G}$ is *removable* if the graph $G' = G \setminus e$ is in $\mathcal{G}$. For a class $\mathcal{H} \subseteq \mathcal{G}$ and a rooted tree $T \in \mathcal{T}$, we define $p(\mathcal{H}, T)$ to be the probability that given a uniform random graph $H \in \mathcal{H}$, and a uniform random pendant copy of T in H, the graph H' obtained by deleting the edge that connects the pendant copy of T to the rest of the graph belongs to $\mathcal{G}$ (and not only to $\mathcal{H}$). In other words, $p(\mathcal{H}, T)$ is the average over all graphs in $\mathcal{H}$ of the proportion of pendant copies of T that are attached using a removable edge. This notion is inspired by bridge-alterable classes, for which $p(\mathcal{H}, T) = 1$, for every $\mathcal{H} \subseteq \mathcal{G}$ and every $T \in \mathcal{T}$ [1, 9]. We do an slight abuse of notation by writing $p(G, T)$ for $p(\{G\}, T)$, for each $G \in \mathcal{G}$. Also, in the cases where $p(G, T)$ is not well-defined (that is, if G has no pendant copy of T), we interpret the probability as 1.

The next theorem says that ζ-tight bridge-addable classes *of graphs* (not only forests) are essentially also bridge-alterable.

▶ **Theorem 13.** *For every θ, there exist a ζ, an n_0 and an ℓ such that for every ζ-tight bridge-addable class $\mathcal{G}$ and $n \geq n_0$, we have that if G_n is a graph chosen uniformly at random in $\mathcal{G}_n$, and v is a vertex chosen uniformly at random in G_n, the following holds with probability at least $1 - \theta$: v is connected to G_n through a removable edge and the corresponding pendant tree has order at most ℓ. In particular, $p(\mathcal{G}_n, T) \geq 1 - \theta$, for every rooted tree $T \in \mathcal{T}_{\leq \ell}$.*

We first sketch how the theorem is proved for classes of forests. Fix $k \geq 1$ and $U_1, \dots, U_k \in \mathcal{U}$. Let $T_1, \dots, T_s$ be the possible rooted versions of U_k. By (8), the ratio between $|\mathcal{G}_n^{k+1,\{U_1,\dots,U_k\}}|$ and $|\mathcal{G}_n^{k,\{U_1,\dots,U_{k-1}\}}|$ is essentially fixed. Moreover, by (9), we know that a typical graph $G \in \mathcal{G}_n^{k,\{U_1,\dots,U_{k-1}\}}$ has $\sum_{i=1}^s \alpha^G(T_i) \approx \sum_{i=1}^s \frac{e^{-|T_i|}}{\mathrm{Aut}_r(T_i)} = \frac{|U_k| e^{-|U_k|}}{\mathrm{Aut}_u(U_k)}$ pendant trees such that, if the edge from where they hang is removable, then they give rise to a graph in $\mathcal{G}_n^{k+1,\{U_1,\dots,U_k\}}$. It turns out that the only way we can realize the desired ratio, is by having almost every such edge removable. Since the choice of k and $U_1, \dots, U_k$ is arbitrary, we are done.

To transfer the result of Theorem 13 from classes of forests to classes of graphs, we use a nice argument introduced in [2]. Every graph admits a unique decomposition into 2-blocks, joined by edges in a tree-like fashion. Consider the partition of $\mathcal{G}_n$ into subclasses $\mathcal{H}_1, \mathcal{H}_2, \dots$ such that every two graphs H and H' in the same subclass, have the same 2-blocks. Since every subclass $\mathcal{H}_i$ is bridge-addable, one can use an averaging argument to show that if $\mathcal{G}$ is ζ-tight, then there exists ζ' such that if n is large enough, then at least $(1 - \zeta')|\mathcal{G}_n|$ graphs are in subclasses $\mathcal{H}_i$ that are ζ'-tight. Let $\mathcal{H}$ be one of such ζ'-tight subclasses of $\mathcal{G}_n$ and let $\mathcal{F}_{\mathcal{H}}$ be the class of forests obtained by selecting the same spanning tree for each 2-block of the graphs in $\mathcal{H}$. Since $\mathcal{F}_{\mathcal{H}}$ is also a ζ'-tight bridge-addable class of forests, we can apply Theorem 13 to it, and the conclusion of the theorem naturally transfers from $\mathcal{F}_{\mathcal{H}}$ to $\mathcal{H}$. Since most of the graphs are in ζ'-tight bridge-addable classes, the statement of Theorem 13 also holds for general classes of graphs. The full proof is presented in [5].

Our next goal is to show that not only the pendant graphs obtained when deleting a removable edge have bounded size, as Theorem 13 ensures, but in fact, they are pendant trees. For every class $\mathcal{G}_n$ and every $t \geq 1$, given G_n chosen uniformly at random from $\mathcal{G}_n$ and v chosen uniformly at random from the vertices of G_n, let $q(\mathcal{G}_n, t)$ be the probability that v is connected to G_n via a removable edge and the corresponding pendant graph is a tree of order at most t. Observe that if $\mathcal{G}$ is subclass of forests, Theorem 13 implies that for every $\theta > 0$, and under some technical conditions, there exists some ℓ such that $q(\mathcal{G}_n, \ell) \geq 1 - \theta$. Next lemma shows that the same holds for general classes of graphs.

▶ **Lemma 14.** *For every $\vartheta > 0$, there exist a ζ, an n_0 and a t, such that if $\mathcal{G}$ is a ζ-tight class and $n \geq n_0$, then $q(\mathcal{G}_n, t) \geq 1 - \vartheta$.*

As before, we split the class $\mathcal{G}_n$ into subclasses $\mathcal{H}_1, \mathcal{H}_2, \ldots$ according to the 2-blocks. Recall that there exists a ζ' such that at least $(1 - \zeta')|\mathcal{G}_n|$ graphs are in subclasses $\mathcal{H}_i$ that are ζ'-tight. Let $\mathcal{H}$ be one of such ζ'-tight subclasses of $\mathcal{G}_n$ and let $\mathcal{F}_{\mathcal{H}}$ the corresponding class of forests. By Theorem 13, if ζ' is small enough and, n and t are large enough (t plays the role of ℓ), then the probability that a random vertex v in a random graph F_n from $\mathcal{F}_{\mathcal{H}}$ connects to F_n through a removable edge and disconnects a pendant tree T_v of order at most t, is close to 1. If this is the case, by construction of $\mathcal{F}_{\mathcal{H}}$, this edge is also a removable cut-edge in the graph in $\mathcal{H}$ that corresponds to F_n.

It remains to show that, with probability close to 1, the pullback H_v of the tree T_v in the original graph in $\mathcal{H}$ is a also tree. This is done by applying Theorem 13 again but using now ℓ much larger than t, which shows that the proportion of vertices that are linked to the rest of the graph by a removable edge is very close to 1, and by noticing that, if H_v is not a tree, then at least one vertex of H_v does not have this property. Details are given in [5].

The last lemma is the key point in proving Theorem 4 for classes that are not forests. Indeed, Theorem 4 now follows relatively easily from Theorem 12 and Lemma 14 (see [5] for a detailed proof).

References

1 Louigi Addario-Berry, Colin McDiarmid, and Bruce Reed. Connectivity for bridge-addable monotone graph classes. *Combin. Probab. Comput.*, 21(6):803–815, 2012.

2 Paul Balister, Béla Bollobás, and Stefanie Gerke. Connectivity of addable graph classes. *J. Combin. Theory Ser. B*, 98(3):577–584, 2008.

3 Itai Benjamini and Oded Schramm. Recurrence of distributional limits of finite planar graphs. *Electron. J. Probab.*, 6:no. 23, 13 pp. (electronic), 2001.

4 Guillaume Chapuy and Guillem Perarnau. Connectivity in bridge-addable graph classes: the McDiarmid-Steger-Welsh conjecture. *Extended abstract in the proceedings of SODA 2016. Long version submitted for publication, see arXiv:1504.06344.*, 2015.

5 Guillaume Chapuy and Guillem Perarnau. Local convergence and stability of tight bridge-addable graph classes. *In preparation.*, 2016.

6 Paul Erdős. On some new inequalities concerning extremal properties of graphs. In *Theory of Graphs (Proc. Colloq., Tihany, 1966)*, pages 77–81, 1966.

7 Paul Erdős. Some recent results on extremal problems in graph theory. *Results, Theory of Graphs (Internat. Sympos., Rome, 1966), Gordon and Breach, New York*, pages 117–123, 1967.

8 Paul Erdős and M Simonovits. A limit theorem in graph theory. In *Studia Sci. Math. Hung.* Citeseer, 1966.

9 Mihyun Kang and Konstantinos Panagiotou. On the connectivity of random graphs from addable classes. *J. Combin. Theory Ser. B*, 103(2):306–312, 2013.

10 László Lovász. *Large networks and graph limits*, volume 60 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, RI, 2012.

11 Colin McDiarmid, Angelika Steger, and Dominic J. A. Welsh. Random graphs from planar and other addable classes. In *Topics in discrete mathematics*, volume 26 of *Algorithms Combin.*, pages 231–246. Springer, Berlin, 2006.

12 Alfréd Rényi. Some remarks on the theory of trees. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 4:73–85, 1959.

13 Miklós Simonovits. A method for solving extremal problems in graph theory, stability problems. In *Theory of Graphs (Proc. Colloq., Tihany, 1966)*, pages 279–319, 1968.

Belief Propagation on Replica Symmetric Random Factor Graph Models*

Amin Coja-Oghlan[1] **and Will Perkins**[2]

1 **Goethe University, Mathematics Institute, Frankfurt, Germany**
`acoghlan@math.uni-frankfurt.de`

2 **School of Mathematics, University of Birmingham, Edgbaston, Birmingham, U.K.**
`math@willperkins.org`

Abstract

According to physics predictions, the free energy of random factor graph models that satisfy a certain "static replica symmetry" condition can be calculated via the Belief Propagation message passing scheme [20]. Here we prove this conjecture for a wide class of random factor graph models. Specifically, we show that the messages constructed just as in the case of acyclic factor graphs asymptotically satisfy the Belief Propagation equations and that the free energy density is given by the Bethe free energy formula.

1998 ACM Subject Classification G.3 Probability and Statistics, F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics

Keywords and phrases Gibbs distributions, Belief Propagation, Bethe Free Energy, Random k-SAT

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.27

1 Introduction and results

1.1 Factor graphs

It is well known that viewing combinatorial optimization problems through the lens of Gibbs measures reveals important information about both structural and algorithmic aspects. For example, suppose that $\Phi = \Phi_1 \wedge \cdots \wedge \Phi_m$ is a k-SAT instance with m clauses over n Boolean variables. We identify the set of all possible truth assignments with the Hamming cube $\{0,1\}^n$, and given a parameter $\beta \geq 0$ we define functions $\psi_i : \{0,1\}^n \to (0,\infty)$ by letting

$$\psi_{\beta,i}(\sigma) = \exp(-\beta\, \mathbf{1}\{\sigma \text{ violates clause } \Phi_i\}). \tag{1.1}$$

These functions induce a probability measure on $\{0,1\}^n$ by letting

$$\mu_{\Phi,\beta} : \sigma \in \{0,1\}^n \mapsto \frac{1}{Z_{\Phi,\beta}} \prod_{i=1}^{m} \psi_{\beta,i}(\sigma), \quad \text{where} \quad Z_{\Phi,\beta} = \sum_{\tau \in \{0,1\}^n} \prod_{i=1}^{m} \psi_{\beta,i}(\sigma)$$

ensures normalization. The measure $\mu_{\Phi,\beta}$ is known as the *Gibbs measure of* Φ *at inverse temperature* β and $Z_{\Phi,\beta}$ is called the *partition function*. Writing out the definition of $\mu_{\Phi,\beta}$,

* The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant Agreement n. 278857–PTCC.

we find

$$\mu_{\Phi,\beta}(\sigma) = \frac{1}{Z_{\Phi,\beta}} \exp(-\beta \cdot \#\,\text{clauses violated under the truth assignment } \sigma).$$

Hence, while $\mu_{\Phi,0}$ is just the uniform distribution over all assignments, as we increase β the probability mass shifts to "more satisfying" assignments. Ultimately, in the limit $\beta \to \infty$ the Gibbs measure concentrates on maximally satisfying assignments. Thus, by tuning β we can scan through the landscape on the Hamming cube defined by the function that maps each truth assignment to the number of clauses it leaves unsatisfied. This landscape has, of course, a very substantial impact on the performance of algorithms. For instance, local search algorithms such as Simulated Annealing are apt to get stuck in local minima. Moreover, the partition function, or equivalently the scaled *free energy* $n^{-1} \ln Z_{\Phi,\beta}$, encompasses important combinatorial characteristics of the optimization problem. For example, the maximum number of clauses that can be satisfied simultaneously equals $m + \lim_{\beta \to \infty} \frac{\partial}{\partial \beta} \ln Z_{\Phi,\beta}$.

Factor graph models provide a general framework for the study of Gibbs measures associated with combinatorial problems [21, 23]. Formally, a *factor graph*, $G = (V(G), F(G), \partial_G, (\psi_a)_{a \in F(G)})$, consists of a finite set $V(G)$ of *variable nodes*, a set $F(G)$ of *constraint nodes* and a function $\partial_G : F(G) \to \bigcup_{l \geq 0} V(G)^l$ that assigns each constraint node $a \in F(G)$ a finite sequence $\partial a = \partial_G a$ of variable nodes, whose length is denoted by $d(a) = d_G(a)$. Additionally, there is a finite set Ω of *spins* and each constraint node $a \in F$ comes with a *weight function* $\psi_a : \Omega^{d(a)} \to (0, \infty)$. The factor graph gives rise to the *Gibbs measure* μ_G on $\Omega^{V(G)}$. Indeed, letting $\sigma(x_1, \ldots, x_k) = (\sigma(x_1), \ldots, \sigma(x_k))$ for $\sigma \in \Omega^{V(G)}$ and $x_1, \ldots, x_k \in V(G)$, we define

$$\mu_G : \sigma \in \Omega^{V(G)} \mapsto \frac{1}{Z_G} \prod_{a \in F(G)} \psi_a(\sigma(\partial a)), \qquad \text{where} \quad Z_G = \sum_{\tau \in \Omega^{V(G)}} \prod_{a \in F(G)} \psi_a(\sigma(\partial a)) \tag{1.2}$$

is the *partition function.* Moreover, G induces a bipartite graph on $V(G) \cup F(G)$ in which the constraint node a is adjacent to the variable nodes that appear in the sequence ∂a. By (slight) abuse of notation we just write $\partial a = \partial_G a$ for the set of such variable nodes. Conversely, for $x \in V(G)$ we let $\partial x = \partial_G x$ be the set of all $a \in F(G)$ such that $x \in \partial a$ and we let $d(x) = d_G(x) = |\partial x|$. The bipartite graph gives rise to a metric on the set of variable and constraint nodes, namely the length of a shortest path.

As we saw above, a k-SAT instance Φ induces a factor graph naturally. Indeed, the variable nodes are just the Boolean variables $x_1, \ldots, x_n$ of the formula Φ and the constraint nodes are the clauses $\Phi_1, \ldots, \Phi_m$. Moreover, $\partial \Phi_i$ is the set of Boolean variables that occur in clause Φ_i, $\Omega = \{0, 1\}$ and the weight functions are given by (1.1).

1.2 Belief Propagation

A fundamental algorithmic task is to calculate the *free energy*, $\ln Z_G$, of a factor graph G. While this is $\#P$-hard in general, in the case that G, viz. the associated bipartite graph, is acyclic the problem can be solved exactly by means of a message passing algorithm called *Belief Propagation* [23].

For a variable node x and an adjacent constraint node a let $\mu_{G,x\to a}$ be the marginal of the spin value of x in the factor graph $G - a$ obtained deleting a. Formally, $\mu_{G,x\to a}(\omega)$ is the probability that x is assigned the spin $\omega \in \Omega$ in a random configuration $\boldsymbol{\sigma} \in \Omega^{V(G)}$ drawn from the Gibbs measure μ_{G-a}. Similarly, let $\mu_{G,a\to x}$ be the marginal of x in the factor graph obtained from G by deleting all constraint nodes $b \in \partial x$, $b \neq a$. We call $\mu_{G,x\to a}$ the *message* from x to a and conversely $\mu_{G,a\to x}$ the message from a to x. If G is acyclic, then for all $x \in V(G)$, $a \in \partial x$, $\sigma \in \Omega$ we have

$$\mu_{G,x\to a}(\sigma) = \frac{\prod_{b\in\partial x}\mu_{G,b\to x}(\sigma)}{\sum_{\tau\in\Omega}\prod_{b\in\partial x}\mu_{G,b\to x}(\tau)}, \tag{1.3}$$
$$\mu_{G,a\to x}(\sigma) = \frac{\sum_{\tau\in\Omega^{\partial a}}\mathbf{1}\{\tau(x)=\sigma\}\psi_a(\tau)\prod_{y\in\partial a\setminus x}\mu_{G,y\to a}(\tau(y))}{\sum_{\tau\in\Omega^{\partial a}}\psi_a(\tau)\prod_{y\in\partial a\setminus x}\mu_{G,y\to a}(\tau(y))}.$$

In fact, the messages $\mu_{G,x\to a}, \mu_{G,a\to x}$ defined above are the unique solution to (1.3). Moreover, these messages can be computed via a fixed point iteration and the number of iterations steps required is bounded by the diameter of G. Furthermore, the *Bethe free energy*, defined as

$$\mathcal{B}_G = \sum_{x\in V(G)}\ln\left[\sum_{\tau\in\Omega}\prod_{b\in\partial x}\mu_{G,b\to x}(\tau)\right] + \sum_{a\in F(G)}\ln\left[\sum_{\tau\in\Omega^{\partial a}}\psi_a(\tau)\prod_{x\in\partial a}\mu_{G,x\to a}(\tau(x))\right] \tag{1.4}$$
$$- \sum_{\substack{a\in F(G)\\ x\in\partial a}}\ln\left[\sum_{\sigma\in\Omega}\mu_{G,a\to x}(\sigma)\mu_{G,x\to a}(\sigma)\right],$$

is equal to $\ln Z_G$. The denominators in (1.3) and the arguments of the logarithms above are positive because of our assumption that the weight functions ψ_a take strictly positive values.

1.3 Random factor graph models

Over the past few years there has been a great deal of interest in the Gibbs measures of random factor graph models. Concrete examples of random factor graph models occur in discrete mathematics and computer science as well as other related areas such as information theory [1, 31]. The following class is already reasonably comprehensive. Let Ω be a finite set of 'spins', let $k \geq 2$ be an integer, let $\Psi \neq \emptyset$ be a finite set of functions $\psi : \Omega^k \to (0,\infty)$ and let $\rho = (\rho_\psi)_{\psi\in\Psi}$ be a probability distribution on Ψ. Then for an integer $n > 0$ and a real $d > 0$ we define the random factor graph $\boldsymbol{G}_n = \boldsymbol{G}_n(d,\Omega,k,\Psi,\rho)$ as follows. The set of variable nodes is $V(\boldsymbol{G}_n) = \{x_1,\ldots,x_n\}$ and the set of constraint nodes is $F(\boldsymbol{G}_n) = \{a_1,\ldots,a_m\}$, where m is a Poisson random variable with mean dn/k. Furthermore, independently for each $i = 1,\ldots,m$ a weight function $\psi_{a_i} \in \Psi$ is chosen from the distribution ρ. Finally, $\partial a_i \in \{x_1,\ldots,x_n\}^k$ is a uniformly random k-tuple of variables, chosen independently for each i. For fixed d,Ω,k,Ψ,ρ, we say the random factor graph $\boldsymbol{G}_n$ has a property $\mathcal{A}$ *asymptotically almost surely* ('a.a.s.') if $\lim_{n\to\infty}\mathrm{P}[\boldsymbol{G}_n \in \mathcal{A}] = 1$.

Much of the recent work on random factor graph models has been guided by ideas from statistical physics. In fact, physicists have developed an analytic but non-rigorous approach to calculating the free energy in random factor graph models, the "cavity method" [23, 24]. The cavity method comes in several installments. The simplest but perhaps most practically important version is called the *replica symmetric ansatz.* It holds that random factor graphs can basically be treated as though they were acyclic: the "messages" defined exactly as in the acyclic case satisfy the Belief Propagation equations (1.3) (at least approximately) and the free energy is given by (1.4) (at least asymptotically).

According to an important physics conjecture the replica symmetric ansatz applies if the random factor graph model enjoys a certain pairwise decorrelation property [20]. Specifically, for a variable node $x \in V(G)$ of a factor graph G we let $\mu_{G,x}$ denote the Gibbs marginal of x. Similarly, we let $\mu_{G,x,y}$ be the joint distribution of the spins assigned to the two variable nodes x, y; thus, $\mu_{G,x,y}$ is the distribution of the pair $(\boldsymbol{\sigma}(x), \boldsymbol{\sigma}(y)) \in \Omega^2$ for $\boldsymbol{\sigma} \in \Omega^{V(G)}$ chosen from the Gibbs measure. Further, let $\|\cdot\|_{\mathrm{TV}}$ denote the total variation norm. Then the replica symmetric solution is conjectured to be correct if

$$\lim_{n\to\infty} \frac{1}{n^2} \sum_{i,j=1}^{n} \mathrm{E} \left\| \mu_{\boldsymbol{G}_n,x_i,x_j} - \mu_{\boldsymbol{G}_n,x_i} \otimes \mu_{\boldsymbol{G}_n,x_j} \right\|_{\mathrm{TV}} = 0. \tag{1.5}$$

In words, a.a.s. the spins assigned to two randomly chosen variable nodes of the random factor graph $\boldsymbol{G}$ are asymptotically independent.

Observe that the distance between two randomly chosen variable nodes x_i, x_j in the random factor graph is $\Omega(\ln n)$ a.a.s. Thus, (1.5) could be interpreted as a (very weak) spatial mixing property.

The main result of this paper proves the conjecture that (1.5) is sufficient to make the "replica symmetric ansatz" work. Following (1.3), for a given factor graph G we call the family of messages $\mu_{G,\,\cdot\,\to\,\cdot} = (\mu_{G,x\to a}, \mu_{G,a\to x})_{x\in V(G),a\in F(G),x\in\partial a}$ an *ε-Belief Propagation fixed point* on G if

$$\frac{1}{n} \sum_{\substack{x\in V(G)\\ a\in\partial x\\ \sigma\in\Omega}} \left| \mu_{G,x\to a}(\sigma) - \frac{\prod_{b\in\partial x\setminus a} \mu_{G,b\to x}(\sigma)}{\sum_{\tau\in\Omega} \prod_{b\in\partial x\setminus a} \mu_{G,b\to x}(\tau)} \right|$$
$$+ \left| \mu_{G,a\to x}(\sigma) - \frac{\sum_{\tau\in\Omega^{\partial a}} \mathbf{1}\{\tau(x)=\sigma\} \psi_a(\tau) \prod_{y\in\partial a\setminus x} \mu_{G,y\to a}(\tau(y))}{\sum_{\tau\in\Omega^{\partial a}} \psi_a(\tau) \prod_{y\in\partial a\setminus x} \mu_{G,y\to a}(\tau(y))} \right| < \varepsilon.$$

Thus, the equations (1.3) hold approximately for almost all pairs $x \in V(G)$, $a \in \partial x$.

▶ **Theorem 1.** *If* (1.5) *holds, then there is a sequence* $(\varepsilon_n)_n \to 0$ *such that* $\mu_{\boldsymbol{G}_n,\,\cdot\,\to\,\cdot}$ *is an* ε_n*-Belief Propagation fixed point a.a.s.*

▶ **Theorem 2.** *If* (1.5) *holds and* $\frac{1}{n}\mathcal{B}_{\boldsymbol{G}_n}$ *converges to a real number* B *in probability, then*

$$\lim_{n\to\infty} \frac{1}{n} \mathrm{E}[\ln Z_{\boldsymbol{G}_n}] = B.$$

Since Belief Propagation equations and the Bethe free energy are conjectured to be incorrect if (1.5) is violated[1] [20], we expect that Theorems 1 and 2 are best possible. While we have phrased the above results for factor graph models of Erdős-Rényi type, they generalize to, e.g., regular factor graph models. The details of this are omitted from this extended abstract but they can be found in the full version [9].

1.4 Non-reconstruction

In physics jargon factor graph models that satisfy (1.5) are called *statically replica symmetric*. An obvious question is how (1.5) can be established in practice. One simple sufficient condition is the notion of non-reconstruction, also known as *dynamic replica symmetry* in physics.

For a factor graph G, a variable node x, an integer $\ell \geq 1$ and a configuration $\sigma \in \Omega^{V(G)}$ we let $\nabla_\ell(G, x, \sigma)$ be the set of all $\tau \in \Omega^{V(G)}$ such that $\tau(y) = \sigma(y)$ for all $y \in V(G)$

[1] Except in the presence of a "global symmetry" like in the Ising model, which could be destroyed by an external field.

whose distance from x exceeds ℓ. The random factor graph $\boldsymbol{G}_n = \boldsymbol{G}_n(d, \Omega, k, \Psi, \rho)$ has the *non-reconstruction property* if

$$\lim_{\ell\to\infty} \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{\sigma\in\Omega^n} \mathrm{E}\left[\mu_{\boldsymbol{G}_n}(\sigma) \left\|\mu_{\boldsymbol{G}_n,x_i} - \mu_{\boldsymbol{G}_n,x_i}[\,\cdot\,|\nabla_\ell(\boldsymbol{G}_n, x_i, \sigma)]\right\|_{\mathrm{TV}}\right] = 0. \qquad (1.6)$$

In words, for large enough ℓ and n the random factor graph has the following property a.a.s. If we pick a variable node x_i uniformly at random and if we pick $\boldsymbol{\sigma}$ randomly from the Gibbs distribution, then the expected difference between the "pure" marginal $\mu_{\boldsymbol{G}_n,x_i}$ of x_i and the marginal of x_i in the conditional distribution given the event $\nabla_\ell(\boldsymbol{G}_n, x_i, \boldsymbol{\sigma})$ diminishes.

▶ **Lemma 3.** *If* (1.6) *holds, then so does* (1.5).

Non-reconstruction is a sufficient but not a necessary condition for (1.5). For instance, in the random graph coloring problem, (1.5) is satisfied in a *much* wider regime of parameters than (1.6) [8, 20, 25].

2 Discussion and related work

The main results of this paper facilitate the "practical" use of Belief Propagation to analyze the free energy in random factor graph models, particularly in combination with Lemma 3. A first example of this kind of approach is the work on the condensation phase transition in the regular k-SAT model [4]. Basically, the recipe is to establish the condition (1.5), e.g., by way of non-reconstruction, and to study Belief Propagation and its fixed points on the random factor graph. Since the random factor graph generally has several Belief Propagation fixed points (unlike in the acyclic case), an extra argument such as an *a priori* bound will be necessary to select the one that yields the actual free energy, cf. [4].

The Belief Propagation fixed point iteration has been used algorithmically on random factor graphs with considerable empirical success (e.g., [19]). Theorem 1 may go as far as one can hope for in terms of a generic explanation of the algorithmic success of Belief Propagation. In fact, the theorem shows that the "true" messages are an asymptotic Belief Propagation fixed point, and the missing piece is to analyze the rate of convergence towards the correct fixed point and its basin of attraction. However, both of these tasks must depend on the specific model.

We always assume that the weight functions ψ_a associated with the constraint nodes are strictly positive: this rules out "hard" constraints. But we impose this condition at least partly out of convenience, namely to ensure that all the quantities that we work with are well-defined, no questions asked. For instance, it is straightforward to extend the present arguments extend to the hard-core model on independent sets.

In an important paper, Dembo and Montanari [12] made progress towards putting the physics predictions on factor graphs, random or not, on a rigorous basis. They proved, inter alia, that a certain "long-range correlation decay" property reminiscent of non-reconstruction is sufficient for the Belief Propagation equations to hold on a certain class of factor graphs whose local neighborhoods converge to trees [12, Theorem 3.14]. Following this, Dembo, Montanari, and Sun [14] verified the Bethe free energy formula for locally tree-like factor graphs under the assumption of Gibbs uniqueness along an interpolating path in parameter space. We contrast non-reconstruction (1.6) to this much stronger uniqueness property which states that the influence of the *worst-case* boundary condition on the marginal spin distribution of x_i decreases in the limit of large ℓ and n.

The present paper builds upon the "regularity lemma" for measures on discrete cubes from [3]. In combinatorics, the "regularity method", which developed out of Szemerédi's regularity lemma for graphs [32], has become an indispensable tool. Bapst and Coja-Oghlan [3] adapted Szemerédi's proof to measures on a discrete cube, such as the Gibbs measure of a (random) factor graph, and showed that this result can be combined with the second moment method to calculate the free energy under certain assumptions. These assumptions are more restrictive than our condition (1.5).

Furthermore, inspired by the theory of graph limits [22], Coja-Oghlan, Perkins and Skubch [10] put forward a "limiting theory" for discrete probability measures to go with the regularity concept from [3]. They applied this concept to random factor graphs under the assumption that (1.5) holds *and* that the Gibbs measures converge in probability (in the topology constructed in [10]). These assumptions are stronger and more complicated to state than (1.5).

Additionally, the present paper builds upon ideas from Panchenko's work [28, 29, 30]. In particular, we follow [28, 29, 30] in using the Aizenman-Sims-Starr scheme [2] to calculate the free energy. Moreover, [29] provides a promising approach towards a general formula for the free energy in Poisson random factor graph models. Specifically, [29] yields a variational formula for the free energy under the assumption that the Gibbs measures satisfies a "finite replica symmetry breaking" condition, which is more general than (1.5). Another assumption of [29] is that the weight functions of the factor graph model must satisfy certain "convexity conditions" to facilitate the use of the interpolation method, which is needed to upper-bound the free energy. By comparison to [29] the main point of the present paper is to justify the Belief Propagation equations, which are at very core of the physicists' "cavity method" in factor graph models, and to obtain a formula for the free energy in terms of the Belief Propagation messages rather than in terms of an abstract variational problem. Practically, the upshot is that by studying the Belief Propagation equations directly on the factor graph we can use geometric clues provided by the graphical structure, as illustrated in [4].

Finally, the proof of Lemma 3 is a fairly straightforward extension of the proof of [10, Proposition 3.4]. That proof, in turn, is a generalization of an argument from [27]. For more on non-reconstruction thresholds in random factor graph models see [7, 11, 16, 25].

3 Proofs of the main results

Here we give an overview of the proofs of the main results. Complete proofs and proofs of the results for random regular factor graphs can be found in the full version of the paper [9]. Throughout this section we fix parameters d, Ω, k, Ψ, ρ of the factor graph model such that (1.5) holds.

3.1 The "cavity trick"

The basic idea behind the physicists' cavity method is to heuristically track the effect of removing a single variable or constraint node from the factor graph, a strategy that is vaguely reminiscent of turning a sampling algorithm into a counting algorithm [18]. The main point of this paper is that we make this heuristic approach rigorous by using the regularity lemma from [3]. Other applications of the cavity method to computing the free energy of Gibbs distributions on lattices include [15].

But before we start let us illustrate the power of this "cavity trick" with an excellent example, the so-called "Aizenman-Simms-Starr scheme" [2], which we are going to use to prove Theorem 2. This is nothing but the following observation. In order to prove that

$\lim_{n\to\infty} n^{-1}\mathrm{E}[\ln Z_{\boldsymbol{G}_n}] = B$ it suffices construct a coupling of the two random factor graphs $\boldsymbol{G}_{n-1}, \boldsymbol{G}_n$ such that

$$\lim_{n\to\infty} \mathrm{E}\left[\ln \frac{Z_{\boldsymbol{G}_n}}{Z_{\boldsymbol{G}_{n-1}}}\right] = B. \tag{3.1}$$

Indeed, since $\mathrm{E}[\ln(Z_{\boldsymbol{G}_n}/Z_{\boldsymbol{G}_{n-1}})] = \mathrm{E}[\ln Z_{\boldsymbol{G}_n}] - \mathrm{E}[\ln Z_{\boldsymbol{G}_{n-1}}]$ and $\ln Z_{\boldsymbol{G}_n} = O(n)$ with certainty, summing up (3.1) yields $\lim_{n\to\infty} n^{-1}\mathrm{E}[\ln Z_{\boldsymbol{G}_n}] = B$. Moreover, as we shall explore in Section 3.3 in detail, we can couple $\boldsymbol{G}_n, \boldsymbol{G}_{n-1}$ by means of a common random super-graph $\hat{\boldsymbol{G}}$ such that $\boldsymbol{G}_n$ is obtained from $\hat{\boldsymbol{G}}$ by removing a few random constraint nodes, while $\boldsymbol{G}_{n-1}$ results from $\hat{\boldsymbol{G}}$ by removing a random variable node along with the adjacent constraint nodes. With this coupling we obtain

$$\mathrm{E}\left[\ln \frac{Z_{\boldsymbol{G}_n}}{Z_{\boldsymbol{G}_{n-1}}}\right] = \mathrm{E}\left[\ln \frac{Z_{\boldsymbol{G}_n}}{Z_{\hat{\boldsymbol{G}}}}\right] - \mathrm{E}\left[\ln \frac{Z_{\boldsymbol{G}_{n-1}}}{Z_{\hat{\boldsymbol{G}}}}\right].$$

Thus, computing the free energy comes down to investigating the impact of removing a few constraint nodes from $\hat{\boldsymbol{G}}$.

To control the effect of such an operation we use two main tools. Both require the following definition. Let $\varepsilon > 0$, let $l \geq 2$ be an integer and let μ be a probability measure on Ω^V for some finite set V. For $x_1, \ldots, x_l \in V$ we write $\mu_{x_1,\ldots,x_l}$ for the joint distribution of random the l-tuple $(\boldsymbol{\sigma}(x_1), \ldots, \boldsymbol{\sigma}(x_l)) \in \Omega^l$ with $\boldsymbol{\sigma}$ chosen from μ. Thus, $\mu_{x_1,\ldots,x_l}$ is the joint distribution of the coordinates $x_1, \ldots, x_l$. Now, we say that μ is (ε, l)*-symmetric* if

$$\sum_{x_1,\ldots,x_l \in V} \|\mu_{x_1,\ldots,x_l} - \mu_{x_1} \otimes \cdots \otimes \mu_{x_l}\|_{\mathrm{TV}} < \varepsilon |V|^l.$$

In words, if we choose coordinates $x_1, \ldots, x_l$ from V randomly, then the expected total variation distance between the joint distribution $\mu_{x_1,\ldots,x_l}$ and the product of the marginals $\mu_{x_1}, \ldots, \mu_{x_l}$ is less than ε. Hence, (1.5) entails that $\mu_{\boldsymbol{G}_n}$ is $(\varepsilon, 2)$-symmetric a.a.s. for any fixed $\varepsilon > 0$. Our first tool is

▶ **Lemma 4.** *For any $\varepsilon > 0, l \geq 3$ there exists $\delta > 0$ such that for all $n > 1/\delta$ and all $\mu \in \mathcal{P}(\Omega^n)$ the following is true:*

If μ is $(\delta, 2)$-symmetric, then μ is (ε, l)-symmetric.

Hence, (1.5) actually implies that $\mu_{\boldsymbol{G}_n}$ is (ε, l)-symmetric a.a.s. for any fixed $\varepsilon > 0$ and any fixed $l \geq 2$. Lemma 4 follows from [3, Corollary 2.3 and 2.4]. (Note that (ε, l)-symmetry is not the same as (approximate) l-wise independence, and so Lemma 4 is not saying that pairwise independence implies l-wise independence).

The second, far more crucial tool is a lemma that allows us to control the effect of adding a few constraints to a factor graph. Specifically, if we make a bounded number of modifications to a factor graph with an $(\varepsilon, 2)$-symmetric Gibbs measure, then the Gibbs measure of the modified graph measure is still $(\alpha, 2)$-symmetric, provided $\varepsilon = \varepsilon(\alpha)$ is small enough. Moreover, the Gibbs marginals remain approximately the same.

▶ **Lemma 5.** *For any integer $L > 0$ and any $\alpha > 0$ there exist $\varepsilon = \varepsilon(\alpha, L) > 0$, $n_0 = n_0(\varepsilon, L)$ such that the following is true. Suppose that G is a factor graph with $n > n_0$ variable nodes such that $\psi_a \in \Psi$ for all $a \in F(G)$. Moreover, assume that μ_G is $(\varepsilon, 2)$-symmetric. If G^+ is obtained from G by adding L constraint nodes $b_1, \ldots, b_L$ with weight functions $\psi_{b_1}, \ldots, \psi_{b_L} \in \Psi$ arbitrarily, then μ_{G^+} is $(\alpha, 2)$-symmetric and*

$$\sum_{x \in V(G)} \left\|\mu_{G,x} - \mu_{G^+,x}\right\|_{\mathrm{TV}} < \alpha n. \tag{3.2}$$

Let us postpone the proof of Lemma 5 to Section 3.4 and instead proceed to derive our main results from Lemma s 4 and 5.

To this end, we need some more notation. We write $\mathcal{P}(\Omega)$ for the set of all probability measures on the finite set Ω, which we identify with the set of all maps $p : \Omega \to [0,1]$ such that $\sum_{\omega\in\Omega} p(\omega) = 1$. If $\mu \in \mathcal{P}(\Omega^S)$ for some finite set $S \neq \emptyset$, then we write $\boldsymbol{\tau}^{\mu}, \boldsymbol{\sigma}^{\mu}, \boldsymbol{\sigma}_1^{\mu}, \boldsymbol{\sigma}_2^{\mu}, \ldots$ for independent samples from μ. We usually omit the superscript. Furthermore, if $X : (\Omega^S)^l \to \mathbb{R}$ is a random variable, then we write

$$\langle X\rangle_{\mu} = \langle X(\boldsymbol{\sigma}_1^{\mu}, \ldots, \boldsymbol{\sigma}_l^{\mu})\rangle_{\mu} = \sum_{\sigma_1,\ldots,\sigma_l\in\Omega^S} X(\sigma_1,\ldots,\sigma_l) \prod_{i=1}^{l} \mu(\sigma_i)$$

for the expectation of X with respect to $\mu^{\otimes l}$. The standard symbols $\mathrm{E}[\,\cdot\,]$, $\mathrm{P}[\,\cdot\,]$ refer to the choice of a random factor graph. Moreover, by default the $O(\,\cdot\,)$-notation refers to the asymptotics as $n \to \infty$.

3.2 The Belief Propagation equations: proof of Theorem 1

The high-level summary of the proof is as follows. Our aim is to verify that typically the Belief Propagation equations (1.3) are approximately satisfied for the message from a random variable node to a random adjacent constraint node and vice versa. Because our factor graph $\boldsymbol{G}_n$ is random, we can prove this claim by way of the "cavity paradigm" as follows. We take a random factor graph $\boldsymbol{G}'$ on $n-1$ variable nodes and add a single variable node and a random set of constraint nodes joining it to the rest of the graph to form the factor graph $\boldsymbol{G}''$. Because the set of variable nodes from $\boldsymbol{G}'$ that are attached to the new constraint nodes are chosen uniformly at random, our assumption (1.5) and Lemma 5 will imply that their messages in $\boldsymbol{G}''$ are approximately the same as their marginals in $\boldsymbol{G}'$, and Lemma 4 will imply that asymptotically the joint distribution of the "attachment points" factorizes a.a.s. Doing the math yields the equations (1.3) plus an $o(1)$-error term.

Let us look now at the details. Given $\varepsilon > 0$ choose $L = L(\varepsilon) > 0$ and $\gamma = \gamma(\varepsilon, L) > \eta = \eta(\gamma) > \delta = \delta(\eta) > 0$ small enough and assume that $n > n_0(\delta)$ is sufficiently large. Because the distribution of the random factor graph $\boldsymbol{G}_n$ is symmetric under permutations of the variable nodes, it suffices to prove that with probability at least $1-\varepsilon$ we have

$$\sum_{a\in\partial x_n,\sigma\in\Omega} \left| \mu_{\boldsymbol{G}_n,x_n\to a}(\sigma) - \frac{\prod_{b\in\partial x\setminus a} \mu_{\boldsymbol{G}_n,b\to x_n}(\sigma)}{\sum_{\tau\in\Omega}\prod_{b\in\partial x_n\setminus a} \mu_{\boldsymbol{G}_n,b\to x_n}(\tau)} \right| < \varepsilon \quad (3.3)$$

and

$$\sum_{a\in\partial x_n,\sigma\in\Omega} \left| \mu_{\boldsymbol{G}_n,a\to x_n}(\sigma) - \frac{\sum_{\tau\in\Omega^{\partial a}} \mathbf{1}\{\tau(x_n)=\sigma\}\psi_a(\tau)\prod_{y\in\partial a\setminus x_n} \mu_{\boldsymbol{G}_n,y\to a}(\tau(y))}{\sum_{\tau\in\Omega^{\partial a}} \psi_a(\tau)\prod_{y\in\partial a\setminus x_n} \mu_{\boldsymbol{G}_n,y\to a}(\tau(y))} \right| < \varepsilon. \quad (3.4)$$

To prove (3.3)–(3.4) let $\boldsymbol{G}'$ be the random factor graph with variable nodes $x_1,\ldots,x_n$ comprising of $\boldsymbol{m}' = \mathrm{Po}(dn(1-1/n)^k/k)$ random constraint nodes $a_1,\ldots,a_{\boldsymbol{m}'}$ that do not contain x_n. Moreover, let $\Delta = \mathrm{Po}(dn(1-(1-1/n)^k)/k)$ be independent of $\boldsymbol{m}'$ and obtain $\boldsymbol{G}''$ from $\boldsymbol{G}'$ by adding independent random constraint nodes $b_1,\ldots,b_\Delta$ with $x_n \in \partial b_i$ for all $i \in [\Delta]$. Since $\boldsymbol{G}''$ has precisely the same distribution as $\boldsymbol{G}_n$, it suffices to verify (3.3)–(3.4) with $\boldsymbol{G}_n$ replaced by $\boldsymbol{G}''$.

Since $dn(1-(1-1/n)^k)/k = d + o(1)$, we can choose $L = L(\varepsilon)$ so large that

$$\mathrm{P}\left[\Delta > L\right] < \varepsilon/3. \quad (3.5)$$

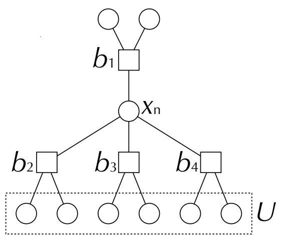


Figure 1 Stitching x_n on to $\boldsymbol{G}'$.

Furthermore, $\boldsymbol{G}'$ is distributed precisely as the random factor graph $\boldsymbol{G}_n$ given that $\partial x_n = \emptyset$. Therefore, Bayes' rule and our assumption (1.5) imply

$$\begin{aligned}\mathrm{P}\left[\boldsymbol{G}' \text{ fails to be } (\delta,2)\text{-symmetric}\right] &\le \mathrm{P}\left[\boldsymbol{G}_n \text{ fails to be } (\delta,2)\text{-symmetric}\right]/\mathrm{P}\left[\partial_{\boldsymbol{G}_n} x_n = \emptyset\right] \\ &\le \exp(d+o(1))\mathrm{P}\left[\boldsymbol{G}_n \text{ fails to be } (\delta,2)\text{-symmetric}\right] < \delta,\end{aligned} \tag{3.6}$$

provided that n_0 is large enough. Combining (3.6) and Lemma 4, we see that

$$\mathrm{P}\left[\boldsymbol{G}' \text{ is } (\eta, 2+(k-1)L)\text{-symmetric}|\Delta \le L\right] > 1-\delta, \tag{3.7}$$

provided δ is sufficiently small.

Due to (3.5) and (3.7) and the symmetry amongst $b_1,\ldots,b_\Delta$ we just need to prove the following: given that $\boldsymbol{G}'$ is $(\eta, 2+(k-1)L)$-symmetric and $0<\Delta\le L$, with probability at least $1-\varepsilon/L$ we have

$$\sum_{\sigma\in\Omega}\left|\mu_{\boldsymbol{G}'',x_n\to b_1}(\sigma) - \frac{\prod_{i=2}^{\Delta}\mu_{\boldsymbol{G}'',b_i\to x_n}(\sigma)}{\sum_{\tau\in\Omega}\prod_{i=2}^{\Delta}\mu_{\boldsymbol{G}'',b_i\to x_n}(\tau)}\right| < \varepsilon/L \tag{3.8}$$

and

$$\sum_{\sigma\in\Omega}\left|\mu_{\boldsymbol{G}'',b_1\to x_n}(\sigma) - \frac{\sum_{\tau\in\Omega^{\partial b_1}}\mathbf{1}\{\tau(x_n)=\sigma\}\psi_{b_1}(\tau)\prod_{y\in\partial b_1\setminus x_n}\mu_{\boldsymbol{G}_n,y\to b_1}(\tau(y))}{\sum_{\tau\in\Omega^{\partial b_1}}\psi_a(\tau)\prod_{y\in\partial b_1\setminus x_n}\mu_{\boldsymbol{G}_n,y\to b_1}(\tau(y))}\right| < \varepsilon/L. \tag{3.9}$$

To this end, let $U=\bigcup_{j\ge 2}\partial b_j$ be the set of all variable nodes that occur in the constraint nodes $b_2,\ldots,b_\Delta$, cf. Figure 1. Because $\mu_{\boldsymbol{G}'',x_n\to b_1}$ is the marginal of x_n in the factor graph $\boldsymbol{G}''-b_1$, the definition (1.2) of the Gibbs measure entails that for any $\sigma\in\Omega$,

$$\begin{aligned}&\mu_{\boldsymbol{G}'',x_n\to b_1}(\sigma) \\ &= \frac{\sum_{\tau\in\Omega^{V(\boldsymbol{G}'')}}\mathbf{1}\{\tau(x_n)=\sigma\}\prod_{a\in F(\boldsymbol{G}')}\psi_a(\tau(\partial a))\prod_{j=2}^{\Delta}\psi_{b_j}(\tau(\partial b_j))}{\sum_{\tau\in\Omega^{V(\boldsymbol{G}'')}}\prod_{a\in F(\boldsymbol{G}')}\psi_a(\tau(\partial a))\prod_{j=2}^{\Delta}\psi_{b_j}(\tau(\partial b_j))} &(3.10)\\ &= \frac{\sum_{\tau\in\Omega^U}\mathbf{1}\{\tau(x_n)=\sigma\}\left\langle\mathbf{1}\{\forall y\in U\setminus\{x_n\}:\boldsymbol{\sigma}(y)=\tau(y)\right\rangle_{\mu_{\boldsymbol{G}'}}\prod_{j=2}^{\Delta}\psi_{b_j}(\tau(\partial b_j))}{\sum_{\tau\in\Omega^U}\left\langle\mathbf{1}\{\forall y\in U\setminus\{x_n\}:\boldsymbol{\sigma}(y)=\tau(y)\right\rangle_{\mu_{\boldsymbol{G}'}}\prod_{j=2}^{\Delta}\psi_{b_j}(\tau(\partial b_j))}. &(3.11)\end{aligned}$$

Similarly, because $\mu_{\boldsymbol{G}'',b_i\to x_n}$ is the marginal of x_n in $\boldsymbol{G}'+b_i$, we have

$$\mu_{\boldsymbol{G}'',b_i\to x_n}(\sigma) = \frac{\sum_{\tau\in\Omega^{\partial b_i}}\mathbf{1}\{\tau(x_n)=\sigma\}\left\langle\mathbf{1}\{\forall y\in\partial b_i\setminus\{x_n\}:\boldsymbol{\sigma}(y)=\tau(y)\right\rangle_{\mu_{\boldsymbol{G}'}}\psi_{b_i}(\tau)}{\sum_{\tau\in\Omega^{\partial b_i}}\left\langle\mathbf{1}\{\forall y\in\partial b_i\setminus\{x_n\}:\boldsymbol{\sigma}(y)=\tau(y)\right\rangle_{\mu_{\boldsymbol{G}'}}\psi_{b_i}(\tau)}. \tag{3.12}$$

To prove (3.8), recall that the variable nodes $\partial b_j \setminus x_n$ are chosen uniformly and independently for each $j \geq 2$. Therefore, if $\boldsymbol{G}'$ is $(\eta, 2+(k-1)L)$-symmetric and $0 < \Delta \leq L$, then

$$\sum_{\tau \in \Omega^U} \mathrm{E}\left[\left|\langle \mathbf{1}\{\forall y \in U \setminus \{x_n\} : \boldsymbol{\sigma}(y) = \tau(y)\rangle_{\mu_{\boldsymbol{G}'}} - \prod_{y \in U} \mu_{\boldsymbol{G}',y}(\tau(y))\right| \,\middle|\, \boldsymbol{G}'\right] \leq 2\eta.$$

Hence, by Markov's inequality, with probability at least $1-\eta^{1/3}$ we have

$$\sum_{\tau \in \Omega^U} \left|\langle \mathbf{1}\{\forall y \in U \setminus \{x_n\} : \boldsymbol{\sigma}(y) = \tau(y)\rangle_{\mu_{\boldsymbol{G}'}} - \prod_{y \in U} \mu_{\boldsymbol{G}',y}(\tau(y))\right| < \eta^{1/3}. \tag{3.13}$$

Set

$$\nu_i(\sigma) = \sum_{\tau \in \Omega^{\partial b_i}} \mathbf{1}\{\tau(x_n) = \sigma\}\psi_{b_i}(\tau) \prod_{y \in \partial b_i \setminus x_n} \mu_{\boldsymbol{G}',y}(\tau(y)). \tag{3.14}$$

A.a.s. for any $1 \leq i < j \leq \Delta$ we have $\partial b_i \cap \partial b_j = \{x_n\}$. Hence, assuming that $\eta = \eta(\gamma) > 0$ is chosen small enough, we obtain from (3.11), (3.12), (3.13) that with probability at least $1-\gamma$,

$$\left|\mu_{\boldsymbol{G}'',x_n\to b_1}(\sigma) - \frac{\prod_{i=2}^{\Delta}\nu_i(\sigma)}{\sum_{\tau\in\Omega}\prod_{i=2}^{\Delta}\nu_i(\tau)}\right| < \gamma \qquad \text{and} \qquad \left|\mu_{\boldsymbol{G}'',b_i\to x_n}(\sigma) - \frac{\nu_i(\sigma)}{\sum_{\tau\in\Omega}\nu_i(\tau)}\right| < \gamma \tag{3.15}$$

for $i \in [\Delta]$. Hence, (3.8) follows from (3.15), provided that γ is chosen small enough.

Finally, to prove (3.9) we use Lemma 5. Let $\boldsymbol{G}''' = \boldsymbol{G}'' - b_1$ be the graph obtained from $\boldsymbol{G}'$ by merely adding $b_2, \dots, b_\Delta$. Given that $\boldsymbol{G}'$ is $(\eta, 2)$-symmetric, Lemma 5 and Lemma 4 imply that $\boldsymbol{G}'''$ is $(\gamma^3, k-1)$-symmetric. As $\partial b_1 \setminus x_n$ is a random subset of size at most $k-1$ chosen independently of $b_2, \dots, b_\Delta$, we conclude that with probability at least $1-\gamma$ over the choice of $\boldsymbol{G}''$,

$$\begin{aligned} 2\gamma &> \sum_{\tau\in\Omega^{\partial b_1}} \left|\langle \mathbf{1}\{\forall y \in \partial b_1 \setminus x_n : \boldsymbol{\sigma}(y) = \tau(y)\}\rangle_{\mu_{\boldsymbol{G}'''}} - \prod_{y\in\partial b_1\setminus x_n} \mu_{\boldsymbol{G}''',y}(\tau(y))\right| \\ &= \sum_{\tau\in\Omega^{\partial b_1}} \left|\langle \mathbf{1}\{\forall y \in \partial b_1 \setminus x_n : \boldsymbol{\sigma}(y) = \tau(y)\}\rangle_{\mu_{\boldsymbol{G}'''}} - \prod_{y\in\partial b_1\setminus x_n} \mu_{\boldsymbol{G}'',y\to b_1}(\tau(y))\right|. \end{aligned} \tag{3.16}$$

Moreover, (3.2) implies that with probability at least $1-\gamma$,

$$2\gamma > \sum_{\tau\in\Omega^{\partial b_1}} \left|\langle \mathbf{1}\{\forall y \in \partial b_1 \setminus x_n : \boldsymbol{\sigma}(y) = \tau(y)\}\rangle_{\mu_{\boldsymbol{G}'''}} - \prod_{y\in\partial b_1\setminus x_n} \mu_{\boldsymbol{G}',y}(\tau(y))\right|. \tag{3.17}$$

Finally, (3.9) follows from (3.14)–(3.17), provided γ is chosen small enough. ◀

3.3 Proof of Theorem 2

To prove (3.1) we will couple the random variables $Z_{\boldsymbol{G}_{n-1}}, Z_{\boldsymbol{G}_n}$ by way of a third random factor graph $\hat{\boldsymbol{G}}$ (a similar coupling was used in [10]). Specifically, let $\hat{\boldsymbol{G}}$ be the random factor graph with variable nodes $V(\hat{\boldsymbol{G}}) = \{x_1, \dots, x_n\}$ obtained by including $\hat{m} = \mathrm{Po}(n\hat{d}/k)$ independent random constraint nodes, where $\hat{d} = d(n/(n-1))^{k-1}$. For each constraint node a of $\hat{\boldsymbol{G}}$ the weight function ψ_a is chosen from the distribution ρ independently.

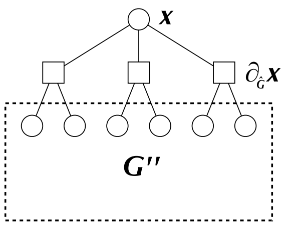


Figure 2 Attaching $\boldsymbol{x}$.

▶ **Lemma 6.** *The two factor graph distributions $\hat{\boldsymbol{G}}, \boldsymbol{G}_n$ have total variation distance $O(1/n)$.*

Proof. The distributions $\mathrm{Po}(dn/k)$, $\mathrm{Po}(\hat{d}n/k)$ have total variation distance $O(1/n)$. ◀

Further, set $p = ((n-1)/n)^{k-1}$ and let $\boldsymbol{G}'$ be a random graph obtained from $\hat{\boldsymbol{G}}$ by deleting each constraint node with probability $1-p$ independently. Let A be the (random) set of constraints removed from $\hat{\boldsymbol{G}}$ to obtain $\boldsymbol{G}'$. In addition, obtain $\boldsymbol{G}''$ from $\hat{\boldsymbol{G}}$ by selecting a variable node $\boldsymbol{x}$ uniformly at random and removing all constraints $a \in \partial_{\hat{\boldsymbol{G}}}\boldsymbol{x}$ along with $\boldsymbol{x}$ itself. Then $\boldsymbol{G}'$ is distributed as $\boldsymbol{G}_n$ and $\boldsymbol{G}''$ is distributed as $\boldsymbol{G}_{n-1}$ plus an isolated variable. Thus,

$$Z_{\boldsymbol{G}_n} \overset{d}{=} Z_{\boldsymbol{G}'}, \qquad Z_{\boldsymbol{G}_{n-1}} \overset{d}{=} Z_{\boldsymbol{G}''}. \tag{3.18}$$

Hence, we are left to calculate $\mathrm{E}[\ln \frac{Z_{\hat{\boldsymbol{G}}}}{Z_{\boldsymbol{G}'}}]$ and $\mathrm{E}[\ln \frac{Z_{\hat{\boldsymbol{G}}}}{Z_{\boldsymbol{G}''}}]$. Much as in the previous proof we will use Lemmas 4 and 5 to trace the effect of tinkering with a small number of constraint nodes. For $x \in V(\hat{\boldsymbol{G}})$, $b \in F(\hat{\boldsymbol{G}})$ we define

$$S_1(x) = \ln\left[\sum_{\sigma\in\Omega}\prod_{a\in\partial_{\hat{\boldsymbol{G}}}x}\mu_{\hat{\boldsymbol{G}},a\to x}(\sigma)\right], \tag{3.19}$$

$$S_2(x) = \sum_{a\in\partial_{\hat{\boldsymbol{G}}}x}\ln\left[\sum_{\tau\in\Omega^{\partial a}}\psi_a(\tau)\prod_{y\in\partial a}\mu_{\hat{\boldsymbol{G}},y\to a}(\tau(y))\right], \tag{3.20}$$

$$S_3(x) = -\sum_{a\in\partial_{\hat{\boldsymbol{G}}}x}\ln\left[\sum_{\tau\in\Omega}\mu_{\hat{\boldsymbol{G}},x\to a}(\tau)\mu_{\hat{\boldsymbol{G}},a\to x}(\tau)\right], \tag{3.21}$$

$$S_4(b) = \ln\left[\sum_{\sigma\in\Omega^{\partial b}}\psi_b(\sigma)\prod_{y\in\partial b}\mu_{\hat{\boldsymbol{G}},y\to b}(\sigma(y))\right]. \tag{3.22}$$

▶ **Lemma 7.** *Let $U = \bigcup_{a\in\partial_{\hat{\boldsymbol{G}}}\boldsymbol{x}}\partial a$. Then a.a.s. we have*

$$\ln\frac{Z_{\hat{\boldsymbol{G}}}}{Z_{\boldsymbol{G}''}} = o(1) + \ln\sum_{\tau\in\Omega^U}\prod_{a\in\partial_{\hat{\boldsymbol{G}}}\boldsymbol{x}}\left[\psi_a(\tau(\partial a))\prod_{y\in\partial a\setminus\boldsymbol{x}}\mu_{\hat{\boldsymbol{G}},y\to a}(\tau(y))\right]. \tag{3.23}$$

Proof. Given $\varepsilon > 0$ let $L = L(\varepsilon) > 0$ be a large enough, let $\gamma = \gamma(\varepsilon, L) > \delta = \delta(\gamma) > 0$ be small enough and assume that n is sufficiently large. Letting $X = |\partial_{\hat{\boldsymbol{G}}}\boldsymbol{x}|$, we can pick L large enough so that

$$\mathrm{P}\left[X > L\right] < \varepsilon. \tag{3.24}$$

As in the previous section, we turn the tables: we think of $\hat{\boldsymbol{G}}$ as being obtained from $\boldsymbol{G}''$ by adding a new variable node $\boldsymbol{x}$ and X independent random constraint nodes $a_1, \ldots, a_X$ such that $x \in \partial a_i$ for all i, cf. Figure 2. The assumption (1.5), Lemma 5 and Lemma 4 imply that

$$\mathrm{P}\left[\sum_{\tau \in \Omega^{U \setminus \{\boldsymbol{x}\}}} \left| \langle \mathbf{1}\{\forall y \in U \setminus \{\boldsymbol{x}\} : \boldsymbol{\sigma}(y) = \tau(y)\}\rangle_{\boldsymbol{G}''} - \prod_{i=1}^{X} \prod_{y \in \partial a_i \setminus \boldsymbol{x}} \mu_{\hat{\boldsymbol{G}}, y \to a_i}(\tau(y)) \right| \geq \delta \middle| X \leq L \right] \\ = o(1). \tag{3.25}$$

Furthermore, unfolding the definition (1.2) of the Gibbs measure, we obtain

$$\frac{Z_{\hat{\boldsymbol{G}}}}{Z_{\boldsymbol{G}''}} = \sum_{\tau \in \Omega^U} \langle \mathbf{1}\{\forall y \in U \setminus \{\boldsymbol{x}\} : \boldsymbol{\sigma}(y) = \tau(y)\}\rangle_{\boldsymbol{G}''} \prod_{i=1}^{X} \psi_{a_i}(\tau(\partial a_i)).$$

Hence, (3.24) and (3.25) show that with probability at least $1 - 2\varepsilon$,

$$\left| \frac{Z_{\hat{\boldsymbol{G}}}}{Z_{\boldsymbol{G}''}} - \sum_{\tau \in \Omega^U} \prod_{i=1}^{X} \left[\psi_{a_i}(\tau(\partial a_i)) \prod_{y \in \partial a_i \setminus \boldsymbol{x}} \mu_{\hat{\boldsymbol{G}}, y \to a_i}(\tau(y)) \right] \right| < \gamma. \tag{3.26}$$

The assertion follows by taking logarithms and sending $\varepsilon \to 0$ slowly as $n \to \infty$. ◀

Combining Lemma 7 with the approximate fixed point property from Theorem 1, we find that (3.23) can be re-formulated as follows.

▶ **Corollary 8.** *A.a.s. we have* $\ln \frac{Z_{\hat{\boldsymbol{G}}}}{Z_{\boldsymbol{G}''}} = S_1(\boldsymbol{x}) + S_2(\boldsymbol{x}) + S_3(\boldsymbol{x}) + o(1)$.

A broadly similar argument yields the following.

▶ **Lemma 9.** *A.a.s. we have* $\ln \frac{Z_{\hat{\boldsymbol{G}}}}{Z_{\boldsymbol{G}'}} = o(1) + \sum_{a \in A} S_4(a)$.

Combining Lemma 9 and Corollary 8, we see that a.a.s. $\hat{\boldsymbol{G}}$ is such that

$$\mathrm{E}\left[\ln \frac{Z_{\boldsymbol{G}'}}{Z_{\boldsymbol{G}''}} \middle| \hat{\boldsymbol{G}}\right] = o(1) + \frac{1}{n}\left[\sum_{x \in V(\hat{\boldsymbol{G}})} (S_1(x) + S_3(x)) + \sum_{a \in F(\hat{\boldsymbol{G}})} S_4(a)\right].$$

Moreover, by our assumption and Fact 6 the r.h.s. converges to B in probability. Thus, Theorem 2 follows by taking the expectation over $\hat{\boldsymbol{G}}$.

3.4 Proof of Lemma 5

The proof of Lemma 5 is based on the "regularity lemma" for probability measures from [3]. Let us introduce the necessary notation. Suppose that $\emptyset \neq U \subset S$ are sets, let $\omega \in \Omega$ and consider $\sigma \in \Omega^S$. Then we let

$$\sigma[\omega|U] = \frac{1}{|U|} \sum_{u \in U} \mathbf{1}\{\sigma(u) = \omega\}.$$

Thus, $\sigma[\,\cdot\,|U] \in \mathcal{P}(\Omega)$ is the distribution of the spin $\sigma(\boldsymbol{u})$ for a uniformly random $\boldsymbol{u} \in U$. Moreover, if $\boldsymbol{V} = (V_1, \ldots, V_l)$ is a partition of some set V, then we call $\#\boldsymbol{V} = l$ the *size* of $\boldsymbol{V}$. Moreover, for $\varepsilon > 0$ we say that $\mu \in \mathcal{P}(\Omega^n)$ is *ε-regular* on a set $U \subset [n]$ if for every subset $S \subset U$ of size $|S| \geq \varepsilon|U|$ we have

$$\langle \|\boldsymbol{\sigma}[\,\cdot\,|S] - \boldsymbol{\sigma}[\,\cdot\,|U]\|_{\mathrm{TV}} \rangle_\mu < \varepsilon.$$

Thus, the empirical distribution of the spins induced on a subset U of S that is "not too small" is typically close to the empirical spin distribution on the entire set S.

Further, μ is *ε-regular* with respect to a partition $\boldsymbol{V}$ if there is a set $J \subset [\#\boldsymbol{V}]$ such that $\sum_{i\in J} |V_i| \geq (1-\varepsilon)n$ and such that μ is ε-regular on V_i for all $i \in J$.

Finally, if $\boldsymbol{V}$ is a partition of $[n]$ and $\boldsymbol{S}$ is a partition of Ω^n, then μ is *ε-homogeneous* w.r.t. $(\boldsymbol{V}, \boldsymbol{S})$ if there is a subset $I \subset [\#\boldsymbol{S}]$ such that the following is true:

HM1: We have $\mu(S_i) > 0$ for all $i \in I$ and $\sum_{i\in I} \mu(S_i) \geq 1-\varepsilon$.

HM2: For all $i \in [\#\boldsymbol{S}]$ and $j \in [\#\boldsymbol{V}]$ we have $\max_{\sigma,\sigma'\in S_i} \|\sigma[\,\cdot\,|V_j] - \sigma'[\,\cdot\,|V_j]\|_{\mathrm{TV}} < \varepsilon$.

HM3: For all $i \in I$ the conditional distribution $\mu[\,\cdot\,|S_i]$ is ε-regular with respect to $\boldsymbol{V}$.

HM4: μ is ε-regular with respect to $\boldsymbol{V}$.

Thus, $\boldsymbol{S}$ is a decomposition of the cube Ω^n such that most of the probability mass belongs to classes S_i such that the conditional measure $\mu[\,\cdot\,|S_i]$ is ε-regular w.r.t. $\boldsymbol{V}$.

▶ **Theorem 10** ([3, Theorem 2.1]). *For any $\varepsilon > 0$ there is an $N = N(\varepsilon) > 0$ such that for every $n > N$, every $\mu \in \mathcal{P}(\Omega^n)$ admits partitions $\boldsymbol{V}$ of $[n]$ and $\boldsymbol{S}$ of Ω^n with $\#\boldsymbol{V} + \#\boldsymbol{S} \leq N$ such that μ is ε-homogeneous with respect to $(\boldsymbol{V}, \boldsymbol{S})$.*

To prove Lemma 5 we look at a partition $(\boldsymbol{V}, \boldsymbol{S})$ as promised by Theorem 10 with respect to which μ_{G^+} is ε-homogeneous. Let $K = \#\boldsymbol{V}$ and $L = \#\boldsymbol{S}$ be such that $K + L \leq N$ and let J be the set of all $j \in [L]$ such that $\mu_{G^+}(S_j) \geq \varepsilon/N$ and $\mu_{G^+}[\,\cdot\,|S_j]$ is ε-regular w.r.t. $\boldsymbol{V}$. Then **HM1** and **HM3** ensure that

$$\sum_{j\notin J} \mu_{G^+}(S_j) < 2\varepsilon. \tag{3.27}$$

Because all functions $\psi \in \Psi$ are strictly positive, we can work out that the original Gibbs measure μ_G is ε'-homogeneous with respect to $(\boldsymbol{V}, \boldsymbol{S})$ as well for some $\varepsilon' > 0$ that depends on ε such that $\varepsilon' \to 0$ as $\varepsilon \to 0$. We then oberve that the $(\varepsilon, 2)$-symmetry of μ_G implies that

$$\sum_{x\in V} \|\mu_{G,x} - \mu_{G,x}[\,\cdot\,|S_j]\|_{\mathrm{TV}} < \varepsilon'' n \tag{3.28}$$

with $\varepsilon'' \to 0$ as $\varepsilon' \to 0$. In other words, the conditional marginals $\mu_{G,x}[\,\cdot\,|S_j]$ induced on the classes S_j are close to the overall marginals $\mu_{G,x}$ for most x. In fact, to derive (3.28) from (1.5) assume that (3.28) were violated. Then μ_G would be a non-trivial mixture of two substantially distinct conditional measures, and it is not difficult to check that this would contradict the $(\varepsilon, 2)$-symmetry of μ_G; the details of this argument are based on results from [3]. Further, **HM2** and (3.28) imply that

$$\sum_{x\in V} \left\|\mu_{G,x} - \mu_{G^+,x}[\,\cdot\,|S_j]\right\|_{\mathrm{TV}} < \varepsilon''' n \tag{3.29}$$

for $j \in J$. Putting the previous argument in reverse, we find that (3.27) and (3.29) imply that μ_{G^+} is $(\alpha, 2)$-symmetric, provided $\varepsilon''' > 0$ was small enough. Additionally, (3.28) and (3.29) imply (3.2). The complete proof of Lemma 5 can be found in the full version of the paper.

Acknowledgements. We thank Florent Krzakala and Lenka Zdeborová for helpful discussions on this topic.

References

1 D. Achlioptas, A. Naor, and Y. Peres: Rigorous location of phase transitions in hard optimization problems. Nature **435** (2005) 759–764.

2 M. Aizenman, R. Sims, S. Starr: An extended variational principle for the SK spin-glass model. Phys. Rev. B **68** (2003) 214403.

3 V. Bapst, A. Coja-Oghlan: Harnessing the Bethe free energy. arXiv:1504.03975 (2015).

4 V. Bapst, A. Coja-Oghlan: The condensation phase transition in the regular k-SAT model. arXiv:1507.03512 (2015).

5 V. Bapst, A. Coja-Oghlan, S. Hetterich, F. Rassmann, D. Vilenchik: The condensation phase transition in random graph coloring. Communications in Mathematical Physics **341** (2016) 543–606.

6 M. Bayati, D. Gamarnik, P. Tetali: Combinatorial approach to the interpolation method and scaling limits in sparse random graphs. Annals of Probability **41** (2013) 4080–4115.

7 N. Bhatnagar, A. Sly, P. Tetali: Reconstruction threshold for the hardcore model. Proc. 14th RANDOM (2010) 434–447.

8 A. Coja-Oghlan, C. Efthymiou, N. Jaafari: Local convergence of random graph colorings. Proc. 19th RANDOM (2015) 726–737.

9 A. Coja-Oghlan, W. Perkins: Belief propagation on replica symmetric random factor graph models. arXiv:1603.08191 (2016).

10 A. Coja-Oghlan, W. Perkins, K. Skubch: Limits of discrete distributions and Gibbs measures on random graphs. arXiv:1512.06798 (2015).

11 C. Efthymiou: Reconstruction/non-reconstruction thresholds for colourings of general Galton-Watson trees. Proc. 19th RANDOM (2015) 756–774.

12 A. Dembo, A. Montanari: Gibbs measures and phase transitions on sparse random graphs. Braz. J. Probab. Stat. **24** (2010) 137–211.

13 A. Dembo, A. Montanari, A. Sly, N. Sun: The replica symmetric solution for Potts models on d-regular graphs. Communications in Mathematical Physics **327** (2014) 551–575.

14 A. Dembo, A. Montanari, N. Sun: Factor models on locally tree-like graphs. Annals of Probability **41** (2013) 4162–4213.

15 D. Gamarnik, D. Katz: Sequential cavity method for computing free energy and surface pressure. Journal of Statistical Physics **137** (2009) 205–232.

16 A. Gerschenfeld, A. Montanari: Reconstruction for models on random graphs. Proc. 48th FOCS (2007) 194–204.

17 S. Janson, T. Łuczak, A. Ruciński: Random Graphs, Wiley 2000.

18 M. Jerrum, L. Valiant, V. Vazirani: Random generation of combinatorial structures from a uniform distribution. Theoretical Computer Science **43** (1986) 169–188.

19 L. Kroc, A. Sabharwal, B. Selman: Message-passing and local heuristics as decimation strategies for satisfiability. Proc 24th SAC (2009) 1408–1414.

20 F. Krzakala, A. Montanari, F. Ricci-Tersenghi, G. Semerjian, L. Zdeborová: Gibbs states and the set of solutions of random constraint satisfaction problems. Proc. National Academy of Sciences **104** (2007) 10318–10323.

21 F. Kschischang, B. Frey, H. Loeliger: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory **47** (2001) 498–519.

22 L. Lovász: Large networks and graph limits. Colloquium Publications **60** (2012), AMS.

23 M. Mézard, A. Montanari: Information, physics and computation. Oxford University Press 2009.

24 M. Mézard, G. Parisi, M. Virasoro: Spin glass theory and beyond. World Scientific 1987.

25 A. Montanari, R. Restrepo, P. Tetali: Reconstruction and clustering in random constraint satisfaction problems. SIAM Journal on Discrete Mathematics **25** (2011) 771–808.

26 C. Moore, S. Mertens: The nature of computation. Oxford University Press (2011).

27 E. Mossel, J. Neeman, A. Sly: Reconstruction and estimation in the planted partition model. Probability Theory and Related Fields (2014) 1–31.

28 D. Panchenko: Spin glass models from the point of view of spin distributions. Annals of Probability **41** (2013) 1315–1361.

29 D. Panchenko: Structure of finite-RSB asymptotic Gibbs measures in the diluted spin glass models. Journal of Statistical Physics **162** (2016) 1–42.

30 D. Panchenko: The Sherrington-Kirkpatrick model. Springer 2013.

31 T. Richardson, R. Urbanke: Modern coding theory. Cambridge University Press (2008).

32 E. Szemerédi: Regular partitions of graphs. Colloq. Internat. CNRS **260** (1978) 399–401.

33 J. Yedidia, W. Freeman, Y. Weiss: Constructing free-energy approximations and generalized Belief Propagation algorithms. IEEE Transactions on Information Theory **51** (2005) 2282–2312.

Towards a Constructive Version of Banaszczyk's Vector Balancing Theorem

Daniel Dadush*[1], Shashwat Garg†[2], Shachar Lovett‡[3], and Aleksandar Nikolov[4]

1 Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
dadush@cwi.nl
2 Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands
s.garg@tue.nl
3 University of California, San Diego, USA
slovett@cs.ucsd.edu
4 University of Toronto, Toronto, Canada
anikolov@cs.toronto.edu

Abstract

An important theorem of Banaszczyk (Random Structures & Algorithms '98) states that for any sequence of vectors of ℓ_2 norm at most $1/5$ and any convex body K of Gaussian measure $1/2$ in $\mathbb{R}^n$, there exists a signed combination of these vectors which lands inside K. A major open problem is to devise a constructive version of Banaszczyk's vector balancing theorem, i.e. to find an efficient algorithm which constructs the signed combination.

We make progress towards this goal along several fronts. As our first contribution, we show an equivalence between Banaszczyk's theorem and the existence of $O(1)$-subgaussian distributions over signed combinations. For the case of symmetric convex bodies, our equivalence implies the existence of a *universal* signing algorithm (i.e. independent of the body), which simply samples from the subgaussian sign distribution and checks to see if the associated combination lands inside the body. For asymmetric convex bodies, we provide a novel *recentering procedure*, which allows us to reduce to the case where the body is symmetric.

As our second main contribution, we show that the above framework can be efficiently implemented when the vectors have length $O(1/\sqrt{\log n})$, recovering Banaszczyk's results under this stronger assumption. More precisely, we use random walk techniques to produce the required $O(1)$-subgaussian signing distributions when the vectors have length $O(1/\sqrt{\log n})$, and use a stochastic gradient ascent method to implement the recentering procedure for asymmetric bodies.

1998 ACM Subject Classification G.3 [Probability and Statistics] Probablistic Algorithms

Keywords and phrases Discrepancy, Vector Balancing, Convex Geometry

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.28

1 Introduction

Given a family of sets $S_1, \dots, S_m$ over a universe $U = [n]$, the goal of combinatorial discrepancy minimization is to find a bi-coloring $\chi : U \to \{-1, 1\}$ such that the discrepancy,

* Supported by the NWO Veni grant 639.071.510.
† Supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 022.005.025.
‡ Supported by an NSF CAREER award 1350481 and a Sloan fellowship.

i.e. the maximum imbalance, $\max_{j\in[m]} |\sum_{i\in S_j} \chi(i)|$ is made as small as possible. Discrepancy theory, where discrepancy minimization plays a major role, has a rich history of applications in computer science as well as mathematics, and we refer the reader to [19, 10, 11] for a general exposition.

A beautiful question regards the discrepancy of sparse set systems, i.e. set systems in which each element appears in at most t sets. A classical theorem of Beck and Fiala [8] gives an upper bound of $2t - 1$ in this setting. They also conjectured an $O(\sqrt{t})$ bound, which if true would be tight. An improved Beck-Fiala bound of $2t - \log^* t$ was given by Bukh [9], where $\log^* t$ is the iterated logarithm function in base 2. Recently, it was shown by Ezra and Lovett [14] that a bound of $O(\sqrt{t \log t})$ holds with high probability when $m \geq n$ and each element is assigned to t sets uniformly at random. The best general bounds having sublinear dependence in t currently depend on n or m. Srinivasan [25] used Beck's *partial coloring method* [7] to give a bound of $O(\sqrt{t} \log \min\{n, m\})$. Using techniques from convex geometry, Banaszczyk [2] proved a general result on vector balancing (stated below) which implies an $O(\sqrt{t \log \min\{n, m\}})$ bound.

The proofs of both Srinivasan's and Banaszczyk's bounds were non-constructive, that is, they provided no efficient algorithm to construct the guaranteed colorings, short of exhaustive enumeration. In the last 6 years, tremendous progress has been made on the question of matching classical discrepancy bounds algorithmically. Currently, essentially all discrepancy bounds proved using the partial coloring method, including Srinivasan's, have been made constructive [4, 18, 15, 22, 13]. Constructive versions of Banaszczyk's result have, however, proven elusive until very recently. In recent work [5], the first and second named authors jointly with Bansal gave a constructive algorithm for recovering Banaszczyk's bound in the Beck-Fiala setting as well as the more general Komlós setting. However, finding a constructive version of Banaszczyk's more general vector balancing theorem, which has further applications in approximating hereditary discrepancy, remains an open problem. This theorem is stated as follows:

▶ **Theorem 1** (Banaszczyk [2]). *Let $v_1, \dots, v_n \in \mathbb{R}^m$ satisfy $\|v_i\|_2 \leq 1/5$. Then for any convex body $K \subseteq \mathbb{R}^m$ of Gaussian measure at least $1/2$, there exists $\chi \in \{-1, 1\}^n$ such that $\sum_{i=1}^n \chi_i v_i \in K$.*

The lower bound $1/2$ on the Gaussian measure of K is easily seen to be tight. In particular, if all the vectors are equal to 0, we must have that $0 \in K$. If we allow Gaussian measure $< 1/2$, then $K = \{x \in \mathbb{R}^n : x_1 \geq \varepsilon\}$, for $\varepsilon > 0$ small enough, is a clear counterexample. On the other hand, it is not hard to see that if K has Gaussian measure $1/2$ then $0 \in K$. Otherwise, there exists a halfspace H containing K but not 0, where H clearly has Gaussian measure less than $1/2$.

Banaszczyk's theorem gives the best known bound for the notorious Komlós conjecture [24], a generalization of the Beck-Fiala conjecture, which states that for any sequence of vectors $v_1, \dots, v_n \in \mathbb{R}^m$ of ℓ_2 norm at most 1, there exists $\chi \in \{-1, 1\}^n$ such that $\|\sum_{i=1}^n \chi_i v_i\|_\infty$ is a constant independent of m and n. In this context, Banaszczyk's theorem gives a bound of $O(\sqrt{\log m})$, because an $O(\sqrt{\log m})$ scaling of the unit ball of ℓ_∞^m has Gaussian measure $1/2$. Banaszczyk's theorem together with estimates on the Gaussian measure of slices of the ℓ_∞^m ball due to Barthe, Guedon, Mendelson, and Naor [6] give a bound of $O(\sqrt{\log d})$, where $d \leq \min\{m, n\}$ is the dimension of the span of $v_1, \dots, v_n$. A well-known reduction (see e.g. Lecture 9 in [24]), shows that this bound for the Komlós problem implies an $O(\sqrt{t \log \min\{m, n\}})$ bound in the Beck-Fiala setting.

While the above results only deal with the case of K being a cube, Banaszczyk's theorem has also been applied to other cases. It was used in [3] to give the best known bound on the Steinitz conjecture. In this problem, the input is a set of vectors $v_1, \dots, v_n$ in $\mathbb{R}^m$ of norm at most one and summing to 0. The aim is to find a permutation $\pi : [n] \to [n]$ to minimise the maximum sum prefix of the vectors rearranged according to π i.e. to minimize $\max_{k \in [n]} \| \sum_{i=1}^{k} v_{\pi(i)} \|$. The Steinitz conjecture is that this bound should always be $O(\sqrt{m})$, irrespective of the number of vectors, and using the vector balancing theorem Banaszczyk proved a bound of $O(\sqrt{m} + \sqrt{\log n})$ for ℓ_2 norm.

More recently, Banaszczyk's theorem was applied to more general symmetric polytopes in Nikolov and Talwar's approximation algorithm [21] for a hereditary notion of discrepancy. Hereditary discrepancy is defined as the maximum discrepancy of any restriction of the set system to a subset of the universe. In [21] it was shown that an efficiently computable quantity, denoted γ_2, bounds hereditary discrepancy from above and from below for any given set system, up to polylogarithmic factors. For the upper bound they used Banaszczyk's theorem for a natural polytope associated with the set system. However, since there is no known algorithmic version of Banaszczyk's theorem for a general body, it is not known how to efficiently compute colorings that achieve the discrepancy upper bounds in terms of γ_2. The recent work on algorithmic bounds in the Komlós setting does not address this more general problem.

Banaszczyk's proof of Theorem 1 follows an ingenious induction argument, which folds the effect of choosing the sign of v_n into the body K. The first observation is that finding a point of the set $\sum_{i=1}^{n} \{-v_i, v_i\}$ inside K is equivalent to finding a point of $\sum_{i=1}^{n-1} \{-v_i, v_i\}$ in $K - v_n \cup K + v_n$. Inducting on this set is not immediately possible because it may no longer be convex. Instead, Banaszczyk shows that a convex subset $K * v_n$ of $(K - v_n) \cup (K + v_n)$ has Gaussian measure at least that of K, as long as K has measure at least $1/2$, which allows him to induct on $K * v_n$. In the base case, he needs to show that a convex body of Gaussian measure at least $1/2$ must contain the origin, but this fact follows easily from the hyperplane separation theorem, as indicated above. While extremely elegant, Banaszczyk's proof can be seen as relatively mysterious as it does not seem to provide any tangible insights as to what the colorings look like.

1.1 Our Results

As our main contribution, we help demystify Banaszczyk's theorem, by showing that it is equivalent, up to a constant factor in the length of the vectors, to the existence of certain subgaussian coloring distributions. Using this equivalence, as our second main contribution, we give an efficient algorithm that recovers Banaszczyk's theorem up to a $O(\sqrt{\log \min\{m, n\}})$ factor for all convex bodies. This improves upon the best previous algorithms of Rothvoss [22], Eldan and Singh [13], which only recover the theorem for symmetric convex bodies up to a $O(\log \min\{m, n\})$ factor.

As a major consequence of our equivalence, we show that for any sequence $v_1, \dots, v_n \in \mathbb{R}^m$ of short enough vectors there exists a probability distribution $\chi \in \{-1, 1\}^n$ over colorings such that, for *any symmetric convex body* $K \subseteq \mathbb{R}^m$ of Gaussian measure at least $1/2$, the random variable $\sum_{i=1}^{n} \chi_i v_i$ lands inside K with probability at least $1/2$. Importantly, if such a distribution can be efficiently sampled, we immediately get a *universal sampler* for constructing Banaszczyk colorings for all symmetric convex bodies (we remark that the recent work of [5] constructs a more restricted form of such distributions). Using random walk techniques, we show how to implement an approximate version of this sampler efficiently, which guarantees the same conclusion when the vectors are of length $O(1/\sqrt{\log \min\{m, n\}})$. We provide more details on these results in Section 2

To extend our results to asymmetric convex bodies, we develop a novel *recentering procedure* and a corresponding efficient implementation which allows us to reduce the asymmetric setting to the symmetric one. After this reduction, a slight extension of the aforementioned sampler again yields the desired colorings. We note that our recentering procedure in fact depends on the target convex body, and hence our algorithms are no longer universal in this setting. We provide more details on these results in Section 3.

Interestingly, we additionally show that this procedure can be extended to yield a completely different coloring algorithm, i.e. not using the sampler, achieving the same $O(\sqrt{\log \min\{m, n\}})$ approximation factor. Surprisingly, the coloring outputted by this procedure is deterministic (its implementation however is not) and has a natural analytic description, which may be of independent interest.

Before we continue with a more detailed description of our results, we begin with some terminology and a well-known reduction. Given a set of vectors $v_1, \ldots, v_n \in \mathbb{R}^m$, we shall call a property *hereditary* if it holds for all subsets of the vectors. We note that Banaszczyk's vector balancing bounds restricted to a set of vectors are hereditary, since a bound on the maximum ℓ_2 norm of the vectors is hereditary. We shall say that a property of colorings holds in the *linear setting*, if when given any shift $t \in \sum_{i=1}^n [-v_i, v_i] \stackrel{\text{def}}{=} \{\sum_{i=1}^n \lambda_i v_i : \lambda \in [-1,1]^n\}$, one can find a coloring (or distribution on colorings) $\chi \in \{-1,1\}^n$ such that $\sum_{i=1}^n \chi_i v_i - t$ satisfies the property. It is well-known that Banaszczyk's theorem also extends by standard arguments to the linear setting after reducing the ℓ_2 norm bound from 1/5 to 1/10 (a factor 2 drop). This follows, for example, from the general inequality between hereditary and linear discrepancy proved by Lovasz, Spencer, and Vesztergombi [16].

All the results in this work will in fact hold in the linear setting. When treating the linear setting, it is well known that one can always reduce to the case where the vectors $v_1, \ldots, v_n$ are linearly independent, and in our setting, when $m = n$. In particular, assume we are given some shift $t \in \sum_{i=1}^n [-v_i, v_i]$ and that $v_1, \ldots, v_n$ are *not* linearly independent. Then, using a standard linear algebraic technique, we can find a "fractional coloring" $x \in [-1,1]^n$ such that $\sum_{i=1}^n x_i v_i = t$, and the vectors $(v_i : i \in A_x)$ are linearly independent, where $A_x \stackrel{\text{def}}{=} \{i : x_i \in (-1,1)\}$ is the set of fractional coordinates (see Lecture 5 in [24], or Chapter 4 in [19]). We can think of this as a reduction to coloring the linearly independent vectors indexed by A_x. Specifically, given x as above, define the lifting function $L_x : [-1,1]^{A_x} \to [-1,1]^n$ by

$$L_x(z)_i = \begin{cases} z_i & : i \in A_x \\ x_i & : i \in [n] \setminus A_x \end{cases}, \ \forall i \in [n] \ . \tag{1}$$

This map takes any coloring $\chi \in \{-1,1\}^{A_x}$ and "lifts" it to a full coloring $L_x(\chi) \in \{-1,1\}^n$. It also satisfies the property that $L_x(\chi) - t = \sum_{i \in A_x} \chi_i v_i - \sum_{i \in A_x} x_i v_i$. So, if we can find a coloring $\chi \in \{-1,1\}^{A_x}$ such that $\sum_{i \in A_x} \chi_i v_i - \sum_{i \in A_x} x_i v_i \in K$, then we would have $L_x(\chi) - t \in K$ as well. Moreover, if we define W as the span of $(v_i : i \in A_x)$, then $\sum_{i \in A_x} \chi_i v_i - \sum_{i \in A_x} x_i v_i \in K$ if and only if $\sum_{i \in A_x} \chi_i v_i - \sum_{i \in A_x} x_i v_i \in K \cap W$, so we can replace K with $K \cap W$, and work entirely inside W. For convex bodies K with Gaussian measure at least 1/2, the central section $K \cap W$ has Gaussian measure that is at least as large, so we have reduced the problem to the case of $|A_x|$ linearly independent vectors in an $|A_x|$-dimensional space (details are given in the full version of this paper.) We shall thus, for simplicity, state all our results in the setting where the vectors $v_1, \ldots, v_n$ are in $\mathbb{R}^n$ and are linearly independent.

2 Symmetric Convex Bodies and Subgaussian Distributions

In this section, we detail the equivalence of Banaszczyk's theorem restricted to symmetric convex bodies with the existence of certain subgaussian distributions. We begin with the main theorem of this section, which we note holds in a more general setting than Banaszczyk's result.

▶ **Theorem 2** (Main Equivalence). *Let $T \subseteq \mathbb{R}^n$ be a finite set. Then, the following parameters are equivalent up to a universal constant factor independent of T and n:*

1. *The minimum $s_b > 0$ such that for any symmetric convex body $K \subseteq \mathbb{R}^n$ of Gaussian measure at least $1/2$, we have that $T \cap s_b K \neq \emptyset$.*
2. *The minimum $s_g > 0$ such that there exists an s_g-subgaussian random variable Y supported on T.*

We recall that a random vector $Y \in \mathbb{R}^n$ is s-subgaussian, or subgaussian with parameter s, if for any unit vector $\theta \in S^{n-1}$ and $t \geq 0$, $\Pr[|\langle Y, \theta\rangle| \geq t] \leq 2e^{-(t/s)^2/2}$. In words, Y is subgaussian if all its 1-dimensional marginals satisfy the same tail bound as the 1-dimensional Gaussian of mean 0 and standard deviation s.

To apply the above to discrepancy, we set $T = \sum_{i=1}^n \{-v_i, v_i\}$, i.e. all signed combinations of the vectors $v_1, \dots, v_n \in \mathbb{R}^n$. In this context, Banaszczyk's theorem directly implies that $s_b \leq 5 \max_{i\in[n]} \|v_i\|_2$, and hence by our equivalence that $s_g = O(1) \max_{i\in[n]} \|v_i\|_2$. Furthermore, the above extends to the linear setting letting $T = \sum_{i=1}^n \{-v_i, v_i\} - t$, for $t \in \sum_{i=1}^n [-v_i, v_i]$, because, as mentioned above, Banaszczyk's theorem extends to this setting as well.

The existence of the *universal sampler* claimed in the previous section is in fact the proof that $s_b = O(s_g)$ in the above Theorem. In particular, it follows directly from the following lemma.

▶ **Lemma 3.** *Let $Y \in \mathbb{R}^n$ be an s-subgaussian random variable. There exists an absolute constant $c > 0$, such for any symmetric convex body $K \subseteq \mathbb{R}^n$ of Gaussian measure at least $1/2$, $\Pr[Y \in s \cdot cK] \geq 1/2$.*

Here, if Y is the s_g-subgaussian distribution supported on $\sum_{i=1}^n \{-v_i, v_i\} - t$ as above, we simply let χ denote the random variable such that $Y = \sum_{i=1}^n \chi_i v_i - t$. That χ now yields the desired universal distribution on colorings is exactly the statement of the lemma.

As a consequence of the above, we see that to recover Banaszczyk's theorem for symmetric convex bodies, it suffices to be able to efficiently sample from an $O(1)$-subgaussian distribution over sets of the type $\sum_{i=1}^n \{-v_i, v_i\} - t$, for $t \in \sum_{i=1}^n [-v_i, v_i]$, when $v_1, \dots, v_n \in \mathbb{R}^n$ are linearly independent and have ℓ_2 norm at most 1. Here we rely on homogeneity, that is, if Y is an s-subgaussian random variable supported on $\sum_{i=1}^n \{-v_i, v_i\} - t$ then αY is αs-subgaussian on $\sum_{i=1}^n \{-\alpha v_i, \alpha v_i\} - \alpha t$, for $\alpha > 0$.

The proof of Lemma 3 follows relatively directly from well-known convex geometric estimates combined with Talagrand's majorizing measure theorem [26] (see also [27]), which gives a powerful characterization of the supremum of any Gaussian process.

Unfortunately, Lemma 3 does not hold for asymmetric convex bodies. In particular, if $Y = -e_1$, the negated first standard basis vector, and $K = \{x \in \mathbb{R}^n : x_1 \geq 0\}$, the conclusion is clearly false no matter how much we scale K, even though Y is $O(1)$-subgaussian and K has Gaussian measure 1/2. One may perhaps hope that the conclusion still holds if we ask for either Y or $-Y$ to be in $s \cdot cK$ in the asymmetric setting, though we do not know how to prove this. We note however that this only makes sense when the support of Y is symmetric, which does not necessarily hold in the linear discrepancy setting.

We now describe the high level idea of the proof for the reverse direction, namely, that $s_g = O(s_b)$. For this purpose, we show that the existence of a $O(s_b)$-subgaussian distribution on T can be expressed as a two player zero-sum game, i.e. the first player chooses a distribution on T and the second player tries to find a non-subgaussian direction. Here the value of the game will be small if and only if the $O(s_b)$-subgaussian distribution exists. To bound the value of the game, we show that an appropriate "convexification" of the space of subgaussianity tests for the second player can be associated with symmetric convex bodies of Gaussian measure at least $1/2$. From here, we use von Neumann's minimax principle to switch the first and second player, and deduce that the value of the game is bounded using the definition of s_b.

2.1 The Random Walk Sampler

From the algorithmic perspective, it turns out that subgaussianity is a very natural property in the context of random walk approaches to discrepancy minimization. Our results can thus be seen as a good justification for the random walk approaches to making Banaszczyk's theorem constructive.

At a high level, in such approaches one runs a random walk over the coordinates of a "fractional coloring" $\chi \in [-1, 1]^n$ until all the coordinates hit either 1 or -1. The steps of such a walk usually come from Gaussian increments (though not necessarily spherical), which try to balance the competing goals of keeping discrepancy low and moving the fractional coloring χ closer to $\{-1, 1\}^n$. Since a sum of small centered Gaussian increments is subgaussian with the appropriate parameter, it is natural to hope that the output of a correctly implemented random walk is subgaussian. Our main result in this setting is that this is indeed possible to a limited extent, with the main caveat being that the walk's output will not be "subgaussian enough" to fully recover Banaszczyk's theorem.

▶ **Theorem 4.** *Let $v_1, \ldots, v_n \in \mathbb{R}^n$ be vectors of ℓ_2 norm at most 1 and let $t \in \sum_{i=1}^n [-v_i, v_i]$. Then, there is an expected polynomial time algorithm which outputs a random coloring $\chi \in \{-1, 1\}^n$ such that the random variable $\sum_{i=1}^n \chi_i v_i - t$ is $O(\sqrt{\log n})$-subgaussian.*

To achieve the above sampler, we guide our random walk using solutions to the so-called vector Kómlos program, whose feasibility was first given by Nikolov [20], and show subgaussianity using well-known martingale concentration bounds. Interestingly, the random walk's analysis does not rely on phases, and is instead based on a simple relation between the walk's convergence time and the subgaussian parameter. As an added bonus, we also give a new and simple constructive proof of the feasibility of the vector Kómlos program which avoids the use of an SDP solver.

Given the results of the previous section, the above random walk is a universal sampler for constructing the following colorings.

▶ **Corollary 5.** *Let $v_1, \ldots, v_n \in \mathbb{R}^n$ be vectors of ℓ_2 norm at most 1, let $t \in \sum_{i=1}^n [-v_i, v_i]$, and let $K \subseteq \mathbb{R}^n$ be a symmetric convex body of Gaussian measure $1/2$ (given by a membership oracle). Then, there is an expected polynomial time algorithm which outputs a coloring $\chi \in \{-1, 1\}^n$ such that $\sum_{i=1}^n \chi_i v_i - t \in O(\sqrt{\log n})K$.*

As mentioned previously, the best previous algorithms in this setting are due to Rothvoss [22], Eldan and Singh [13], which find a signed combination inside $O(\log n)K$. Furthermore, these algorithms are not universal, i.e. they heavily depend on the body K. We note that these algorithms are in fact tailored to find *partial colorings* inside a symmetric convex body

K of Gaussian measure at least 2^{-cn}, for $c > 0$ small enough, a setting in which our sampler does not provide any guarantees.

We now recall prior work on random walk based discrepancy minimization. The random walk approach was pioneered by Bansal [4], who used a semidefinite program to guide the walk and gave the first efficient algorithm matching the classic $O(\sqrt{n})$ bound of Spencer [23] for the combinatorial discrepancy of set systems satisfying $m = O(n)$. Later, Lovett and Meka [18] provided a greatly simplified walk, removing the need for the semidefinite program, which recovered the full power of Beck's entropy method for constructing partial colorings. Harvey, Schwartz, and Singh [15] defined another random walk based algorithm, which, unlike previous work and similarly to our algorithm, doesn't explicitly use phases or produce partial colorings. The random walks of [18] and [15] both depend on the convex body K; the walk in [18] is only well-defined in a polytope, while the one in [15] remains well-defined in any convex body, although the analysis still applies only to the polyhedral setting. Most directly related to this paper is the recent work [5], which gives a walk that can be viewed as a randomized variant of the original $2t-1$ Beck-Fiala proof. This walk induces a distribution $\chi \in \{-1,1\}^n$ on colorings for which *each coordinate* of the output $\sum_{i=1}^n \chi_i v_i$ is $O(1)$-subgaussian. From the discrepancy perspective, this gives a sampler which finds colorings inside any axis parallel box of Gaussian measure at least $1/2$ (and their rotations, though not in a universal manner), matching Banaszczyk's result for this class of convex bodies.

3 Asymmetric Convex Bodies

In this section, we explain how our techniques extend to the asymmetric setting. The main difficulty in the asymmetric setting is that one cannot hope to increase the Gaussian mass of an asymmetric convex body by simply scaling it. In particular, if we take $K \subseteq \mathbb{R}^n$ to be a halfspace through the origin, e.g. $\{x \in \mathbb{R}^n : x_1 \geq 0\}$, then K has Gaussian measure exactly $1/2$ but $sK = K$ for all $s > 0$. At a technical level, the lack of any measure increase under scaling breaks the proof of Lemma 3, which is crucial for showing that subgaussian coloring distributions produce combinations that land inside K.

The main idea to circumvent this problem will be to reduce to a setting where the mass of K is "symmetrically distributed" about the origin, in particular, when the barycenter of K under the induced Gaussian measure is at the origin. For such a body K, we show that a constant factor scaling of $K \cap -K$ also has Gaussian mass at least $1/2$, yielding a direct reduction to the symmetric setting.

To achieve this reduction, we will use a novel *recentering procedure*, which will both carefully fix certain coordinates of the coloring as well as shift the body K to make its mass more "symmetrically distributed". The guarantees of this procedure are stated below:

▶ **Theorem 6** (Recentering Procedure). *Let $v_1, \dots, v_n \in \mathbb{R}^n$ be linearly independent, $t \in \sum_{i=1}^n [-v_i, v_i]$, and $K \subseteq \mathbb{R}^n$ be a convex body of Gaussian measure at least $1/2$. Then, there exists a fractional coloring $x \in [-1,1]^n$, such that for $p = \sum_{i=1}^n x_i v_i - t$, $A_x = \{i \in [n] : x_i \in (-1,1)\}$ and $W = \operatorname{span}(v_i : i \in A_x)$, the following holds:*
1. *$p \in K$.*
2. *The Gaussian measure of $(K-p) \cap W$ on W is at least the Gaussian measure of K.*
3. *The barycenter of $(K-p) \cap W$ is at the origin, i.e. $\int_{(K-p)\cap W} y e^{-\|y\|^2/2} dy = 0$.*

By convention, if the procedure returns a full coloring $x \in \{-1,1\}^n$ (in which case, since $p \in K$, we are done), we shall treat conditions 2 and 3 as satisfied, even though $W = \{0\}$. At a high level, the recentering procedure allows us to reduce the initial vector balancing problem to one in a possibly lower dimension with respect to "well-centered" convex body of no smaller Gaussian measure, and in particular, of Gaussian measure at least 1/2. Interestingly, as mentioned earlier in the introduction, the recentering procedure can also be extended to yield a full coloring algorithm. We explain the high level details of its implementation together with this extension in the next subsection.

To explain how to use the fractional coloring x from Theorem 6 to get a useful reduction, recall the lifting function $L_x : [-1,1]^{A_x} \to [-1,1]^n$ defined in (1). We reduce the initial vector balancing problem to the problem of finding a coloring $\chi \in \{-1,1\}^{A_x}$ such that $\sum_{i \in A_x} \chi_i v_i - \sum_{i \in A_x} x_i v_i \in (K-p) \cap W$ (note that $\sum_{i \in A_x} \chi_i v_i - \sum_{i \in A_x} x_i v_i \in W$ by construction). Then we can lift this coloring to $L_x(\chi)$, which satisfies

$$\sum_{i \in A_x} \chi_i v_i - \sum_{i \in A_x} x_i v_i \in (K-p) \cap W \Leftrightarrow \sum_{i=1}^{n} L_x(\chi)_i v_i - t \in K.$$

From here, the guarantee that $K' \stackrel{\text{def}}{=} (K-p) \cap W$ has Gaussian measure at least 1/2 and barycenter at the origin allows a direct reduction to the symmetric setting. Namely, we can replace K' by the symmetric convex body $K' \cap -K'$ without losing "too much" of the Gaussian measure of K'. This is formalized by the following extension of Lemma 3, which directly implies a reduction to subgaussian sampling as in section 2.

▶ **Lemma 7.** *Let $Y \in \mathbb{R}^n$ be an s-subgaussian random variable. There exists an absolute constant $c > 0$, such for any convex body $K \subseteq \mathbb{R}^n$ of Gaussian measure at least $1/2$ and barycenter at the origin, $\Pr[Y \in s \cdot c(K \cap -K)] \geq 1/2$.*

In particular, if there exists a distribution over colorings $\chi \in \{-1,1\}^{A_x}$ such that $\sum_{i \in A_x} \chi_i v_i - \sum_{i \in A_x} x_i v_i$ as above is $1/c$-subgaussian, Lemma 7 implies that the random signed combination lands inside K' with probability at least 1/2. Thus, the asymmetric setting can be effectively reduced to the symmetric one, as claimed.

Crucially, the recentering procedure in Theorem 6 can be implemented in probabilistic polynomial time if one relaxes the barycenter condition from being exactly 0 to having "small" norm. Furthermore, the estimate in Lemma 7 will be robust to such perturbations. Thus, to constructively recover the colorings in the asymmetric setting, it will still suffice to be able to generate good subgaussian coloring distributions.

Combining the sampler from Theorem 4 together with the recentering procedure, we constructively recover Banaszczyk's theorem for general convex bodies up to a $O(\sqrt{\log n})$ factor.

▶ **Theorem 8** (Weak Constructive Banaszczyk)**.** *There exists a probabilistic polynomial time algorithm which, on input a linearly independent set of vectors $v_1, \ldots, v_n \in \mathbb{R}^n$ of ℓ_2 norm at most $c/\sqrt{\log n}$, $c > 0$ small enough, $t \in \sum_{i=1}^n [-v_i, v_i]$, and a (not necessarily symmetric) convex body $K \subseteq \mathbb{R}^n$ of Gaussian measure at least $1/2$ (given by a membership oracle), computes a coloring $\chi \in \{-1,1\}^n$ such that with high probability $\sum_{i=1}^n \chi_i v_i - t \in K$.*

As far as we are aware, the above theorem gives the first algorithm to recover Banaszczyk's result for asymmetric convex bodies under any non-trivial restriction. In this context, we note that the algorithm of Eldan and Singh [13] finds "relaxed" partial colorings, i.e. where the fractional coordinates of the coloring are allowed to fall outside $[-1,1]$, and the resulting vector lies inside an n-dimensional convex body of Gaussian measure at least 2^{-cn}. However, it is unclear how one could use such partial colorings to recover the above result, even with a larger approximation factor.

3.1 The Recentering Procedure

In this section, we describe the details of the recentering procedure as well as its extension to full colorings, which produces deterministic colorings matching the guarantees of Theorem 8. We provide only its abstract instantiation here, leaving a detailed description of its implementation to the full version of the paper.

Before we begin, we give a more geometric view of the vector balancing problem and the recentering procedure, which help clarify the exposition. Let $v_1, \dots, v_n \in \mathbb{R}^n$ be linearly independent vectors and $t \in \sum_{i=1}^n [-v_i, v_i]$. Given the target body $K \subseteq \mathbb{R}^n$ of Gaussian measure at least $1/2$, we can restate the vector balancing problem geometrically as that of finding a vertex of the parallelepiped $P = \sum_{i=1}^n [-v_i, v_i] - t$ lying inside K. Here, the choice of t ensures that $0 \in P$. Note that this condition is necessary, since otherwise there exists a halfspace separating P from 0 having Gaussian measure at least $1/2$.

Recall now that in the linear setting, and using this geometric language, Banaszczyk's theorem implies that if P contains the origin, and $\max_{i \in [n]} \|v_i\|_2 \leq 1/10$ (which we do not need to assume for the validity of the recentering procedure), then any convex body of Gaussian measure at least $1/2$ contains a vertex of P. Thus, for our given target body K, we should make our situation better by replacing P and K by $P - q$ and $K - q$ respectively, if $q \in P$ is a shift such that $K - q$ has higher Gaussian measure than K. In particular, given the symmetry of Gaussian measure, one would intuitively expect that if the Gaussian mass of K is not symmetrically distributed around 0, there should be a shift of K which increases its Gaussian measure.

In the current language, fixing a color $\chi_i \in \{-1, 1\}$ for vector v_i, corresponds to restricting ourselves to finding a vertex in the facet $F = \chi_i v_i + \sum_{j \neq i} [-v_j, v_j] - t$ of P lying inside K. Again intuitively, restricting to a facet of P should improve our situation if the Gaussian measure of the corresponding slice of K in the lower dimension is larger than that of K. To make this formal, note that when inducting on a facet F of P (which is an $n-1$ dimensional parallelepiped), we must choose a center $q \in F$ to serve as the new origin in the lower dimensional space. Precisely, this can be expressed as inducting on the parallelepiped $F - q$ and shifted slice $(K - q) \cap \mathrm{span}(F - q)$ of K, using the $n - 1$ dimensional Gaussian measure on $\mathrm{span}(F - q)$.

With the above viewpoint, one can restate the goal of the recentering procedure as that of finding a point $q \in P \cap K$, such that smallest face F of P containing q, satisfies that $(K - q) \cap \mathrm{span}(F - q)$ has its barycenter at the origin and Gaussian measure no smaller than that of K. Recall that as long as $(K - q) \cap \mathrm{span}(F - q)$ has Gaussian measure at least $1/2$, we are guaranteed that $0 \in K - q \Rightarrow q \in K$. With this geometry in mind, we implement the recentering procedure as follows:

Compute $q \in P$ so that the Gaussian mass of $K - q$ is maximized. If q is on the boundary of P, letting F denote a facet of P containing q, induct on $F - q$ and the slice $(K - q) \cap \mathrm{span}(F - q)$ as above. If q is in the interior of P, replace P and K by $P - q$ and $K - q$, and terminate.

We now explain why the above achieves the desired result. First, if the maximizer q is in a face F of P, then a standard convex geometric argument reveals that the Gaussian measure of $(K - q) \cap \mathrm{span}(F - q)$ is no smaller than that of $K - q$, and in particular, no smaller than that of K. Thus, in this case, the recentering procedure fixes a color for “free”. In the second case, if q is in the interior of P, then a variational argument gives that the barycenter of $K - q$ under the induced Gaussian measure must be at the origin, namely, $\int_{K-q} x e^{-x^2/2} dx = 0$.

To conclude this section, we explain how to extend the recentering procedure to directly produce a deterministic coloring satisfying Theorem 8. For this purpose, we shall assume that

$v_1, \ldots, v_n$ have length at most $c/\sqrt{\log n}$, for a small enough constant $c > 0$. To begin, we run the recentering procedure as above, which returns P and K, with K having its barycenter at the origin. We now replace P, K by a joint scaling $\alpha P, \alpha K$, for $\alpha > 0$ a large enough constant, so that αK has Gaussian mass at least $3/4$. At this point, we run the original recentering procedure again with the following modification: every time we get to the situation where K has its barycenter at the origin, induct on the facet of P closest to the origin. More precisely, in this situation, compute a point p on the boundary of P closest to the origin, and, letting F denote the facet containing p, induct on $F - p$ and $(K - p) \cap \operatorname{span}(F - p)$. At the end, return the final found vertex.

Notice that, as claimed, the coloring (i.e. vertex) returned by the algorithm is indeed deterministic. The reason the above algorithm works is the following. While we cannot guarantee, as in the original recentering procedure, that the Gaussian mass of $(K - p) \cap \operatorname{span}(F - p)$ does not decrease, we can instead show that it decreases only *very slowly.* In particular, we use the bound of $O(1/\sqrt{\log n})$ on the length of the vectors $v_1, \ldots, v_n$ to show that every time we induct, the Gaussian mass drops by at most a $1 - c/n$ factor. More generally, if the vectors had length at most $d > 0$, for d small enough, the drop would be of the order $1 - ce^{-1/(cd)^2}$, for some constant $c > 0$. Since we "massage" K to have Gaussian mass at least $3/4$ before applying the modified recentering algorithm, this indeed allows to induct n times while keeping the Gaussian mass above $1/2$, which guarantees that the final vertex is in K. To derive the bound on the rate of decrease of Gaussian mass, we prove a new inequality on the Gaussian mass of sections of a convex body near the barycenter, which may be of independent interest. The new inequality is stated below:

► **Theorem 9.** *Let $K \subseteq \mathbb{R}^n$ be a convex body with Gaussian measure $\gamma_n(K) = \alpha \geq 3/5$ such that its barycenter, b satisfies $\|b\|_2 \leq \eta$, for $\eta > 0$ small enough. For $\theta \in S^{n-1}$ and $a \in \mathbb{R}$, let $K_a^\theta = (K - a\theta) \cap \{x \in \mathbb{R}^n : \langle \theta, x \rangle = 0\}$. Then, there exists universal constants $a_0, c > 0$, such that for $|a| \leq a_0$, we have that*

$$\gamma_{n-1}(K_a^\theta) \geq (\alpha - c\eta)(1 - \frac{e^{-\frac{1}{100a^2}}}{4\sqrt{2\pi}}).$$

As a final remark, we note that unlike the subgaussian sampler, the recentering procedure is not scale invariant. Namely, if we jointly scale P and K by some factor α, the output of the recentering procedure will not be an α-scaling of the output on the original K and P, as Gaussian measure is not homogeneous under scalings. Thus, one must take care to appropriately normalize P and K before applying the recentering procedure to achieve the desired results.

We now give the high level overview of our recentering step implementation. The first crucial observation in this context, is that the task of finding $t \in P$ maximizing the Gaussian measure of $K - t$ is in fact a *convex program.* More precisely, the objective function (Gaussian measure of $K - t$) is a logconcave function of t and the feasible region P is convex. Hence, one can hope to apply standard convex optimization techniques to find the desired maximizer.

It turns out however, that one can significantly simplify the required task by noting that the recentering strategy does not in fact necessarily need an exact maximizer, or even a maximizer in P. To see this, note that if p is a shift such that $K - p$ has larger Gaussian measure than K, then by logconcavity the shifts $K - \alpha p$, $0 < \alpha \leq 1$, also have larger Gaussian measure. Thus, if a we find a shift $p \notin P$ with larger Gaussian measure, letting αp be the intersection point with the boundary ∂P, we can induct on the facet of $P - \alpha p$ containing 0 and the corresponding slice of $K - \alpha p$ just as before. Given this, we can essentially "ignore" the constraint $p \in P$ and the treat the optimization problem as unconstrained.

This last observation will allow us to use the following simple gradient ascent strategy. Precisely, we simply take steps in the direction of the gradient until either we pass through a facet of P or the gradient becomes "too small". As alluded to previously, the gradient will exactly equal a fixed scaling of the barycenter of $K - p$, p the current shift, under the induced Gaussian measure. Thus, once the gradient is small, the barycenter will be very close to the origin, which will be good enough for our purposes. The last nontrivial technical detail is how to efficiently estimate the barycenter, where we note that the barycenter is the expectation of a random point inside $K - p$. For this purpose, we simply take an average of random samples from $K - p$, where we generate the samples using standard random walk samplers for logconcave distributions over convex bodies [1, 17, 12].

Conclusion and Open Problems

In conclusion, we have shown a tight connection between the existence of subgaussian coloring distributions and Banaszczyk's vector balancing theorem. Furthermore, we make use of this connection to constructively recover a weaker version of this theorem. The main open problem we leave is thus to fully recover Banaszczyk's result. As explained above, this reduces to finding a distribution on colorings such that the output random signed combination is $O(1)$-subgaussian, when the input vectors have ℓ_2 norm at most 1. We believe this approach is both attractive and feasible, especially given the recent work [5], which builds a distribution on colorings for which each coordinate of the output random signed combination is $O(1)$-subgaussian.

Acknowledgments. We would like to thank the American Institute for Mathematics for hosting a recent workshop on discrepancy theory, where some of this work was done.

References

1 David Applegate and Ravi Kannan. Sampling and integration of near log-concave functions. In *Proceedings of the 23rd annual ACM symposium on Theory of Computing*, pages 156–163, 1991.

2 Wojciech Banaszczyk. Balancing vectors and Gaussian measures of n-dimensional convex bodies. *Random Structures Algorithms*, 12(4):351–360, 1998.

3 Wojciech Banaszczyk. On series of signed vectors and their rearrangements. *Random Struct. Algorithms*, 40(3):301–316, 2012. `doi:10.1002/rsa.20373`.

4 Nikhil Bansal. Constructive algorithms for discrepancy minimization. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science FOCS 2010*, pages 3–10. IEEE Computer Soc., Los Alamitos, CA, 2010.

5 Nikhil Bansal, Daniel Dadush, and Shashwat Garg. An algorithm for Komlós conjecture matching Banaszczyk's bound. To appears in FOCS, 2016.

6 Franck Barthe, Olivier Guédon, Shahar Mendelson, and Assaf Naor. A probabilistic approach to the geometry of the l_p^n-ball. *Ann. Probab.*, 33(2):480–513, 2005.

7 József Beck. Roth's estimate of the discrepancy of integer sequences is nearly sharp. *Combinatorica*, 1(4):319–325, 1981.

8 József Beck and Tibor Fiala. "Integer-making" theorems. *Discrete Appl. Math.*, 3(1):1–8, 1981.

9 Boris Bukh. An improvement of the Beck-Fiala theorem. *CoRR*, abs/1306.6081, 2013.

10 Bernard Chazelle. *The discrepancy method*. Cambridge University Press, Cambridge, 2000. Randomness and complexity.

11 William Chen, Anand Srivastav, Giancarlo Travaglini, et al. *A Panorama of Discrepancy Theory*, volume 2107. Springer, 2014.

12 Ben Cousins and Santosh Vempala. A cubic algorithm for computing Gaussian volume. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1215–1228. ACM, New York, 2014.

13 Ronen Eldan and Mohit Singh. Efficient algorithms for discrepancy minimization in convex sets. CoRR, abs/1409.2913, 2014.

14 Esther Ezra and Shachar Lovett. On the Beck-Fiala conjecture for random set systems. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:190, 2015.

15 Nicholas J. A. Harvey, Roy Schwartz, and Mohit Singh. Discrepancy without partial colorings. In *Approximation, randomization, and combinatorial optimization*, volume 28 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 258–273. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2014.

16 L. Lovász, J. Spencer, and K. Vesztergombi. Discrepancy of set-systems and matrices. *European J. Combin.*, 7(2):151–160, 1986. doi:10.1016/S0195-6698(86)80041-5.

17 László Lovász and Santosh Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 57–68, 2006.

18 Shachar Lovett and Raghu Meka. Constructive discrepancy minimization by walking on the edges. *SIAM J. Comput.*, 44(5):1573–1582, 2015. Preliminary version in FOCS 2012.

19 Jiří Matoušek. *Geometric discrepancy*, volume 18 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1999. An illustrated guide.

20 Aleksandar Nikolov. The Komlós conjecture holds for vector colorings. arXiv preprint arXiv:1301.4039, 2013.

21 Aleksandar Nikolov and Kunal Talwar. Approximating hereditary discrepancy via small width ellipsoids. In *Symposium on Discrete Algorithms, SODA*, pages 324–336, 2015.

22 Thomas Rothvoss. Constructive discrepancy minimization for convex sets. In *55th Annual IEEE Symposium on Foundations of Computer Science – FOCS 2014*, pages 140–145. IEEE Computer Soc., Los Alamitos, CA, 2014.

23 Joel Spencer. Six standard deviations suffice. *Trans. Amer. Math. Soc.*, 289(2):679–706, 1985.

24 Joel Spencer. *Ten lectures on the probabilistic method*, volume 52. SIAM, 1987.

25 Aravind Srinivasan. Improving the discrepancy bound for sparse matrices: better approximations for sparse lattice approximation problems. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (New Orleans, LA, 1997)*, pages 692–701. ACM, New York, 1997.

26 Michel Talagrand. Regularity of Gaussian processes. *Acta Math.*, 159(1-2):99–149, 1987. doi:10.1007/BF02392556.

27 Michel Talagrand. *Upper and lower bounds for stochastic processes*, volume 60 of *Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge. A Series of Modern Surveys in Mathematics [Results in Mathematics and Related Areas. 3rd Series. A Series of Modern Surveys in Mathematics]*. Springer, Heidelberg, 2014. Modern methods and classical problems. doi:10.1007/978-3-642-54075-2.

On the Beck-Fiala Conjecture for Random Set Systems

Esther Ezra[*1] **and Shachar Lovett**[†2]

1 **School of Mathematics, Georgia Institute of Technology, Atlanta, USA**
`eezra3@math.gatech.edu`
2 **Computer Science and Engineering, University of California, San Diego, USA**
`slovett@cse.ucsd.edu`

Abstract

Motivated by the Beck-Fiala conjecture, we study discrepancy bounds for random sparse set systems. Concretely, these are set systems (X, Σ), where each element $x \in X$ lies in t randomly selected sets of Σ, where t is an integer parameter. We provide new bounds in two regimes of parameters. We show that when $|\Sigma| \geq |X|$ the hereditary discrepancy of (X, Σ) is with high probability $O(\sqrt{t \log t})$; and when $|X| \gg |\Sigma|^t$ the hereditary discrepancy of (X, Σ) is with high probability $O(1)$. The first bound combines the Lovász Local Lemma with a new argument based on partial matchings; the second follows from an analysis of the lattice spanned by sparse vectors.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Discrepancy theory, Beck-Fiala conjecture, Random set systems

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.59

1 Introduction

Let (X, Σ) be a finite set system, with X a finite set and Σ a collection of subsets of X. A *two-coloring* of X is a mapping $\chi : X \to \{-1, +1\}$. For a subset $S \in \Sigma$ we define $\chi(S) := \sum_{x \in S} \chi(x)$. The *discrepancy* of Σ is defined as

$$\operatorname{disc}(\Sigma) := \min_{\chi} \max_{S \in \Sigma} |\chi(S)|.$$

In other words, the discrepancy of the set system (X, Σ) is the minimum over all colorings χ of the largest deviation from an even split, over all subsets in Σ. For background on discrepancy theory, we refer the reader to the books of Chazelle [7] and Matoušek [11].

In this paper, our interest is in the discrepancy of sparse set systems. The set system (X, Σ) is said to be t-sparse if any element $x \in X$ belongs to at most t sets $S \in \Sigma$. A well-known result of Beck and Fiala [4] is that sparse set systems have discrepancy bounded only in terms of their sparsity.

▶ **Theorem 1** ([4]). *If* (X, Σ) *is* t*-sparse then* $\operatorname{disc}(\Sigma) \leq 2t - 2$.

The bound was improved to $2t - 3$ by Bednarchak and Helm [5], to $2t - 4$ by Helm [10], and to $2t - \log^* t$ by Bukh [6]. However, Beck and Fiala conjectured that in fact, the correct bound should be $O(\sqrt{t})$, analogous to Spencer's theorem for non-sparse set systems [14]. This is a long standing open problem in discrepancy theory. The best result to date (which allows dependency on the size of the set system) is by Banaszczyk [2].

[*] E.E is supported by an NSF CAREER award 1553354.
[†] S.L. is supported by an NSF CAREER award 1350481 and a Sloan fellowship.

▶ **Theorem 2** ([2]). *If (X, Σ) is t-sparse with $|X| = n$ then* $\operatorname{disc}(\Sigma) \le O(\sqrt{t \log n})$.

Recently, Bansal et al. [3] gave an efficient algorithm which finds a coloring matching Banaszczyk's bound.

1.1 Our results

In this paper, we study random sparse set systems. To sample a random t-sparse set system (X, Σ) with $|X| = n, |\Sigma| = m$, for each $x \in X$ choose uniformly and independently a subset $T_x \subset [m]$ of size $|T_x| = t$. Then set $S_i = \{x \in X : i \in T_x\}$ and $\Sigma = \{S_1, \dots, S_m\}$. Letting $\mathbb{E}[\cdot]$ denote expectation, our main quantity of interest is $\mathbb{E}[\operatorname{disc}(\Sigma)]$. We show that when $m \ge n$, this is close to the conjectured bound of Beck and Fiala. Specifically, we show $\mathbb{E}[\operatorname{disc}(\Sigma)] = O(\sqrt{t \log t})$. In particular, the bound does not depend on n.

In fact, we obtain such bound for the hereditary discrepancy of the set system. For $Y \subset X$ let $\Sigma|_Y = \{S \cap Y : S \in \Sigma\}$ be the set system restricted to Y. The hereditary discrepancy of a set system (X, Σ) is defined as

$$\operatorname{herdisc}(\Sigma) = \max_{Y \subset X} \operatorname{disc}(\Sigma|_Y).$$

Our main result is the following.

▶ **Theorem 3.** *Assume $m \ge n \ge t$. Let (X, Σ) be a random t-sparse set system with $|X| = n, |\Sigma| = m$. Then*

$$\mathbb{E}[\operatorname{disc}(\Sigma)] \le \mathbb{E}[\operatorname{herdisc}(\Sigma)] \le O(\sqrt{t \log t}).$$

In fact, the bound holds with probability $1 - \exp(-\Omega(t))$.

We note that our technique can be extended to the case where $m \ge cn$ for any absolute constant $c > 0$, but fails whenever $m \ll n$. The main reason is that in this regime, most sets are large. Nevertheless, when n is considerably larger than m, we use a different approach and show that the discrepancy is small in this case as well. Specifically, when n is somewhat larger than $\binom{m}{t}$ we show that the discrepancy is only $O(1)$.

▶ **Theorem 4.** *Fix $m \ge t$ and let $N = \binom{m}{t}$. Assume that $n \ge \Omega(N \log N)$. Let (X, Σ) be a random t-sparse set system with $|X| = n, |\Sigma| = m$. Then*

$$\mathbb{E}[\operatorname{disc}(\Sigma)] = O(1).$$

In fact, the bound holds with probability $1 - N^{-\Omega(1)}$.

To summarize, the work in this paper was motivated by the elusive Beck-Fiala conjecture. We considered a natural setting of random t-sparse set systems, and showed that in this case, in some regimes of parameters, the conjecture holds (with the bound of $O(\sqrt{t})$ replaced by the slightly weaker bound of $O(\sqrt{t \log t})$ in our first result). We hope that the techniques developed in this work will be useful for the study of random sparse set systems in the full spectrum of parameters, as well as for the original Beck-Fiala conjecture.

2 Preliminaries and Proof Overview

The Lovász Local Lemma [9] is a powerful probabilistic tool. In this paper we only need its symmetric version.

▶ **Theorem 5.** *Let $E_1, E_2, ..., E_k$ be a series of events such that each event occurs with probability at most p and such that each event is independent of all the other events except for at most d of them. If $ep(d+1) \leq 1$ then* $\Pr[\wedge_{i=1}^{m} \overline{E_i}] > 0$.

In our analysis we exploit a few standard tail bounds for the sum of independent random variables (Chernoff-Hoeffding bounds, see, e.g., [1]).

▶ **Lemma 6** (Tail bounds for additive error). *Let $Z_1, \ldots, Z_k \in \{-1, 1\}$ be independent random variables and let $Z = Z_1 + \ldots + Z_k$. Then for any $\lambda > 0$*

$$\Pr\left[|Z - \mathbb{E}[Z]| \geq \lambda\sqrt{k}\right] \leq 2\exp(-2\lambda^2).$$

▶ **Lemma 7** (Tail bounds for multiplicative errors). *Let $Z_1, \ldots, Z_k \in \{0, 1\}$ be independent random variables and let $Z = Z_1 + \ldots + Z_k$. Then for any $\lambda > 0$*

$$\Pr\left[Z \geq (1+\lambda)\mathbb{E}[Z]\right] \leq \exp(-\lambda^2/3 \cdot \mathbb{E}[Z]).$$

2.1 Proof Overview for Theorem 3

We next present an overview of our proof for Theorem 3. For simplicity of exposition, we present the overview only for the derivation of the discrepancy bound. In Section 3 we present the actual analysis and show a bound on the hereditary discrepancy.

First, we classify each set as being either "small" if its cardinality is $O(t)$, or "large" otherwise. Then we proceed in several steps:

(i) **Making large sets pairwise disjoint:** Initially, we show that with high probability over the choice of the set system, it is possible to delete at most one element from each large set, such that they become pairwise disjoint after the deletion. This property is proved in Lemma 8.

(ii) **Partial matching:** For each large set resulting after step (i), we pair its elements, leaving at most, say, two unpaired elements. Since each pair appears in a unique set, this process results in a *partial matching* $M = \{(a_1, b_1), \ldots, (a_k, b_k)\}$ on X. We observe that as soon as we have such a matching, we can restrict the two-coloring function χ on X to assign *alternating signs* on each pair of M. Since each large set S has at most two unpaired elements, we immediately conclude that $|\chi(S)| \leq 2$.

(iii) **Applying the Lovász Local Lemma on the small sets:** We are thus left to handle the small sets. In this case, we observe that a random coloring χ, with alternating signs on M as above[1], satisfies with positive probability that $|\chi(S)| \leq O(\sqrt{t \log t})$ for all small sets $S \in \Sigma$. This is a consequence of the Lovász Local Lemma, as each small set S contains only $O(t)$ elements, and each of these elements participates in t sets of Σ. The fact that some of these elements appear in the partial matching implies that S can "influence" (w.r.t. the random coloring χ) at most $2|S|t = O(t^2)$ other small sets; see Section 3 for the details.

We point out that as soon as we have a partial matching M as above, we can "neutralize" the deviation that might be caused by the large sets, and only need to keep the deviation, caused by the small sets, small. The latter is fairly standard to do, and so the main effort in the analysis is to show that we can indeed make large sets disjoint as in step (i).

We note that our proof technique is constructive. Our arguments for steps (i) and (ii) (see Lemma 8 and our charging scheme in Claim 10) give an efficient algorithm to find an element

[1] That is, each pair in M is assigned $(+1, -1)$ or $(-1, +1)$ independently with probability 1/2.

to delete in each large set, thereby making large sets disjoint, as well as build the partial matching, or, alternatively, report (with small probability) that a partial matching of the above kind does not exist and halt. In step (iii) we can apply the algorithmic Lovász Local Lemma of Moser and Tardos [12, 13], since the colors are assigned independently among the pairs in M as well as the unpaired elements. Thus, we obtain an expected polynomial time algorithm, which, with high probability over the choice of the set system, constructs a coloring with discrepancy $O(\sqrt{t \log t})$.

3 A Low Hereditary Discrepancy Bound: The Analysis

We now proceed with the proof of Theorem 3. We classify the sets in Σ based on their size. A set $S \in \Sigma$ is said to be *large* if $|S| \geq 6t$ and *small* otherwise. Note that as $m \geq n$, most sets in Σ are small. Let $I = \{i : S_i \text{ is large}\}$ be a random variable capturing the indices of the large sets. To construct a coloring, we proceed in several steps. First, we show that with high probability the large sets are nearly disjoint. We will assume throughout that t is sufficiently large (concretely $t \geq 55$).

▶ **Lemma 8.** *Fix $t \geq 55$. Let E denote the following event: "there exists a choice of $x_i \in S_i$ for $i \in I$ such that the sets $\{S_i \setminus \{x_i\} : i \in I\}$ are pairwise disjoint". Then* $\Pr[E] \geq 1 - 2^{-t}$.

We defer the proof of Lemma 8 to Section 4 and prove Theorem 3 based on it, in the remainder of this section. Decompose

$$\begin{aligned}\mathbb{E}[\text{herdisc}(\Sigma)] &= \mathbb{E}[\text{herdisc}(\Sigma)|E]\Pr[E] + \mathbb{E}[\text{herdisc}(\Sigma)|\overline{E}]\Pr[\overline{E}]\\ &\leq \mathbb{E}[\text{herdisc}(\Sigma)|E] + (2t-1)\Pr[\overline{E}]\\ &\leq \mathbb{E}[\text{herdisc}(\Sigma)|E] + 1\end{aligned}$$

where we bounded $\mathbb{E}[\text{herdisc}(\Sigma)|\overline{E}]$ by the Beck-Fiala theorem (Theorem 1) which holds for any t-sparse set system, and bounded $\Pr[\overline{E}]$ by 2^{-t} according to Lemma 8. To conclude the proof we will show that when E holds then $\text{herdisc}(\Sigma) \leq O(\sqrt{t \log t})$. Thus, we assume from now on that the event E holds. Fix a subset $Y \subset X$, where we will construct a two-coloring for $\Sigma' = \Sigma|_Y$ of low discrepancy.

Partition each $S_i \cap Y = A_i \cup B_i$ for $i \in I$, where $|A_i|$ is even, $|B_i| \leq 2$ and the sets $\{A_i : i \in I\}$ are pairwise disjoint. Partition each A_i arbitrarily into $|A_i|/2$ pairs, and let M be the union of these pairs. That is, M is a partial matching on Y given by $M = \{(a_1, b_1), \ldots, (a_k, b_k)\}$ where $a_1, b_1, \ldots, a_k, b_k \in Y$ are distinct, and each A_i is a union of a subset of M, and each pair a_j, b_j appears in a unique set A_i due to the fact that these sets are pairwise disjoint (they thus form a partition of M). We say that a coloring $\chi : Y \to \{-1, +1\}$ is *consistent with* M if $\chi(a_j) = -\chi(b_j)$ for all $j \in [k]$. Note that if S_i is a large set, then for any coloring χ consistent with M,

$$|\chi(S_i \cap Y)| = |\chi(A_i) + \chi(B_i)| = |0 + \chi(B_i)| \leq |B_i| \leq 2.$$

Thus, we only need to minimize the discrepancy of χ over the small sets in Σ. To do so, we choose χ uniformly from all two-colorings consistent with M. These are given by choosing uniformly and independently $\chi(a_i) \in \{-1, +1\}$ for $i \in [k]$, setting $\chi(b_i) = -\chi(a_i)$ and choosing $\chi(x) \in \{-1, +1\}$ uniformly and independently for all $x \notin \{a_1, b_1, \ldots, a_k, b_k\}$.

Let S_i be a small set, that is $|S_i| \leq 6t$. Let E_i denote the event

$$E_i := \left[|\chi(S_i \cap Y)| \geq c\sqrt{t \log t}\right].$$

Each pair $\{a_j, b_j\}$ contained in S_i contributes 0 to the discrepancy, and all other elements obtain independent colors. Hence $\chi(S_i)$ is the sum of $t' \le 6t$ independent signs. By Lemma 6, for an appropriate constant c we have

$$\Pr[E_i] \le 1/100t^2.$$

We next claim that each event E_i depends on at most $d = 12t^2$ other events $\{E_j : j \neq i\}$. Indeed, let $S'_i = S_i \cup \{a_j : b_j \in S_i\} \cup \{b_j : a_j \in S_i\}$. Then $|S'_i| \le 2|S_i| \le 12t$ and $\chi(S_i)$ is independent of $\chi(x)$ for all $x \notin S'_i$. So, if E_i depends on E_j, it must be the case that S_j intersects S'_i. However, as each $x \in S'_i$ is contained in t sets, there are at most $12t^2$ such events E_j.

We are now in a position to apply the Lovász Local Lemma (Theorem 5). Its condition are satisfied as we have $p = 1/100t^2$ and $d = 12t^2$. Hence $\Pr[\wedge \overline{E_i}] > 0$, that is, there exists a coloring χ consistent with M for which $|\chi(S_i)| \le c\sqrt{t \log t}$ for all small sets S_i. This coloring shows that $\mathrm{disc}(\Sigma') \le \max(c\sqrt{t \log t}, 2)$ as claimed.

4 Proof of Lemma 8

Let (X, Σ) be a t-sparse set system with $|X| = n, |\Sigma| = m$. It will be convenient to identify it with a bi-partite graph $G = (X, V, E)$ where $|V| = m$ and $E = \{(x, i) : x \in S_i\}$. Then, a random t-sparse set system is the same as a random left t-regular bi-partite graph. That is, a uniform graph satisfying $\deg(x) = t$ for all $x \in X$.

Large sets in Σ correspond to the subset of the vertices $V' = \{v \in V : \deg(v) \ge 6t\}$. For a vertex $v \in V$ let $\Gamma(v) \subset X$ denote its neighbors. Lemma 8 is equivalent to the following lemma, which we prove in this section.

▶ **Lemma 9.** *Fix $t \ge 55$. With probability at least $1 - 2^{-t}$ over the choice of G, there exists a choice of $x_v \in \Gamma(v)$ such that the sets $\{\Gamma(v) \setminus \{x_v\} : v \in V'\}$ are pairwise disjoint.*

Let G' be the induced (bi-partite) sub-graph on (X, V'). We will show that with high probability G' has no cycles. In such a case Lemma 9 follows from the straightforward scheme described below:

▶ **Claim 10.** *Assume that G' has no cycles. Then there exists a choice of $x_v \in \Gamma(v)$ such that the sets $\{\Gamma(v) \setminus \{x_v\} : v \in V'\}$ are pairwise disjoint.*

Proof. We present a charging scheme of the vertices $x_v \in \Gamma(v)$, for each $v \in V'$. If G' has no cycles then it is a forest. Fix a tree T in G' and an arbitrary root $v_T \in V'$ of T. Orient the edges of T from v_T to the leaves. For each $v \in T$ other than the root, choose x_v to be the parent of v in the tree, and choose x_{v_T} arbitrarily. Let $A_v = \Gamma(v) \setminus \{x_v\}$ for $v \in V'$. We claim that $\{A_v : v \in V'\}$ are pairwise disjoint. To see that, assume towards contradiction that $x \in A_{v_1} \cap A_{v_2}$ for some $x \in X, v_1, v_2 \in V'$. Then v_1, x, v_2 is a path in G' and hence v_1, v_2 must belong to the same tree T. However, the only case where this can happen (as T is a tree) is that x is the parent of both v_1, v_2 in T. However, by construction in this case $x = x_{v_1} = x_{v_2}$ and hence $x \notin A_{v_1}, A_{v_2}$, from which we conclude that $\{A_v : v \in V'\}$ are pairwise disjoint, as claimed. ◀

In the remainder of the proof we show that with high probability G' has no cycles. The girth of G', denoted $\mathrm{girth}(G')$, is the minimal length of a cycle in G' if such exists, and otherwise it is ∞. Note that as G' is bipartite, then $\mathrm{girth}(G')$ is (if finite) the minimal 2ℓ such that there exist a cycle $x_1, v_1, x_2, v_2, \ldots, x_\ell, v_\ell, x_1$ in G' with $x_i \in X$ and $v_i \in V'$.

▶ **Claim 11.** $\Pr[\mathrm{girth}(G') = 4] \leq t^4 \exp(-t)$.

Proof. Fix $x_1, x_2 \in X$ and $v_1, v_2 \in V$. They form a cycle of length 4 if $v_1, v_2 \in \Gamma(x_1) \cap \Gamma(x_2)$. As each $\Gamma(x_i)$ is a uniformly chosen set of size t we have that

$$\Pr[v_1, v_2 \in \Gamma(x_1) \cap \Gamma(x_2)] = \left(\frac{\binom{t}{2}}{\binom{m}{2}} \right)^2 \leq (t/m)^4.$$

Next, conditioned on the event that $v_1, v_2 \in \Gamma(x_1) \cap \Gamma(x_2)$, we still need to have $v_1, v_2 \in V'$ (that is v_1, v_2 represent large sets of Σ). We will only require that $v_1 \in V'$ for the bound. Note that so far we only fixed $\Gamma(x_1), \Gamma(x_2)$, and hence the neighbors of $\Gamma(x)$ for $x \neq x_1, x_2$ are still uniform. Then $v_1 \in V'$ if at least $6t-2$ other nodes $x \in X$ have v_1 as their neighbor. By Lemma 7, the probability for this is bounded by

$$\Pr[v_1 \in V' | v_1, v_2 \in \Gamma(x_1) \cap \Gamma(x_2)] \leq \exp(-((5t-2)/t)^2/3 \cdot t) \leq \exp(-t).$$

So,

$$\Pr[v_1, v_2 \in \Gamma(x_1) \cap \Gamma(x_2) \wedge v_1 \in V'] \leq (t/m)^4 \cdot \exp(-t).$$

To bound $\Pr[\mathrm{girth}(G') = 4]$ we union bound over all $\binom{n}{2}\binom{m}{2}$ choices of x_1, x_2, v_1, v_2. Using our assumption that $m \geq n$ we get

$$\Pr[\mathrm{girth}(G') = 4] \leq m^4 (t/m)^4 \exp(-t) \leq t^4 \exp(-t). \quad ◀$$

▶ **Claim 12.** *For any* $\ell \geq 3$, $\Pr[\mathrm{girth}(G') = 2\ell] \leq \exp(-t\ell)$.

Proof. Let $x_1, v_1, \ldots, x_\ell, v_\ell$ denote a potential cycle of length 2ℓ. As it is a minimal cycle and $\ell \geq 3$, the vertices v_i, v_j have no common neighbors, unless $j = i+1$ in which case x_i is the only common neighbor of v_i, v_{i+1} (where indices are taken modulo ℓ). Thus there exist sets $X_i \subset X$ of size $|X_i| = 6t-2$ such that $X_i \subset \Gamma(v_i)$ and $X_1, \ldots, X_\ell, \{x_1, \ldots, x_\ell\}$ are pairwise disjoint.

Let $E(x_1, v_1, \ldots, x_\ell, v_\ell, X_1, \ldots, X_\ell)$ denote the event described above, for a fixed choice of $x_1, v_1, \ldots, x_\ell, v_\ell, X_1, \ldots, X_\ell$. The event holds if

1. v_i, v_{i+1} are neighbors of x_i.
2. v_i is a neighbor of all $x \in X_i$.

There are independent events, as $\Gamma(x)$ is independently chosen for each $x \in X$. So

$$\begin{aligned}
&\Pr[E(x_1, v_1, \ldots, x_\ell, v_\ell, X_1, \ldots, X_\ell)] \\
&= \prod_{i=1}^{\ell} \Pr[v_i, v_{i+1} \in \Gamma(x_i)] \cdot \prod_{i=1}^{\ell} \prod_{x \in X_i} \Pr[v_i \in \Gamma(x)] \\
&= \left(\frac{\binom{t}{2}}{\binom{m}{2}} \right)^{\ell} \cdot \left(\frac{t}{m} \right)^{(6t-2)\ell} \leq \left(\frac{t}{m} \right)^{6t\ell}.
\end{aligned}$$

To bound $\Pr[\mathrm{girth}(G') = 2\ell]$ we union bound over all choices of $x_1, v_1, \ldots, x_\ell, v_\ell, X_1, \ldots, X_\ell$. The number of choices is bounded by

$$n^\ell m^\ell \binom{n}{6t-2}^{\ell} \leq \left(\frac{nm \cdot e^{6t-2} \cdot n^{6t-2}}{(6t-2)^{6t-2}} \right)^{\ell} \leq \left(\frac{(em)^{6t}}{(6t-2)^{6t-2}} \right)^{\ell}.$$

Thus,

$$\Pr[\mathrm{girth}(G') = 2\ell] \leq \left(\frac{(em)^{6t}}{(6t-2)^{6t-2}} \right)^{\ell} \cdot \left(\frac{t}{m} \right)^{6t\ell} = \left((6t-2)^2 \left(\frac{et}{6t-2} \right)^{6t} \right)^{\ell} \leq \exp(-t\ell). \quad ◀$$

Proof of Lemma 9. Using Claims 11 and 12, the probability that $\text{girth}(G') < \infty$ is bounded by:

$$\Pr[\text{girth}(G') < \infty] = \sum_{\ell=2}^{\infty} \Pr[\text{girth}(G') = \ell] \le t^4 \exp(-t) + \sum_{\ell=3}^{\infty} \exp(-t\ell) \le 2t^4 \exp(-t).$$

For $t \ge 55$, we have that $\Pr[\text{girth}(G') < \infty] \le 2^{-t}$. ◀

5 The regime of large sets

We next prove Theorem 4. Let (X, Σ) be a t-sparse set system with $|X| = n, |\Sigma| = m$. In this setting, we consider the case of fixed m, t and $n \to \infty$. Consider its $m \times n$ incidence matrix. The columns are t-sparse vectors in $\{0,1\}^m$, and hence have $N = \binom{m}{t}$ possible values. When $n \gg N$, there will be many repeated columns. We show that in this case, the discrepancy of the set system is low. Setting notations, let $v_1, \dots, v_N \in \{0,1\}^m$ be all the possible t-sparse vectors, and let $r_1, \dots, r_N$ denote their multiplicity in the set system. Note that they define the set system uniquely (up to permutation of the columns, which does not effect the discrepancy).

Our main result in this section is the following. We will assume throughout that m is large enough and that $4 \le t \le m-4$. We note that if $t \le 3$ or $t \ge m-3$ then result immediately follows from the Beck-Fiala theorem (Theorem 1), for any set systems. The first case follows by a direct application, and the second case by first partitioning the columns to pairs and subtracting one vector from the next in each pair, which gives a 6-sparse $\{-1, 0, 1\}$ matrix, to which we apply the Beck-Fiala theorem.

▶ **Theorem 13.** *Let (X, Σ) be a t-sparse set system with $4 \le t \le m-4$ and m large enough. Assume that $\min(r_1, \dots, r_N) \ge 7$. Then $\text{disc}(\Sigma) \le 2$.*

Note that the statement in Theorem 13 is somewhat stronger than that in Theorem 4, as it only assumes that all possible t-sparse column vectors comprise the incidence matrix of (X, Σ), and their multiplicity is 7 or higher. In fact, Theorem 4 follows from Theorem 13 using a straightforward coupon-collector argument [8]. In this regime, with high probability (say, with probability at least $1 - 1/N$), a random sample of $\Theta(N \log N)$ columns guarantees that each t-sparse column appears with multiplicity 7 (or higher). Therefore, we obtain:

$$\mathbb{E}[\text{disc}(\Sigma)] \le 2\left(1 - \frac{1}{N}\right) + \frac{2t-1}{N} = O(1).$$

We are thus left to prove Theorem 13. First, we present an overview of the proof.

5.1 Proof overview

Every column v_i is repeated r_i times. As we may choose arbitrary signs for each occurrence of a vector, the aggregate total would be $c_i v_i$, where $c_i \in \mathbb{Z}$, $|c_i| \le r_i$ and $c_i \equiv r_i \mod 2$. Our goal is to show that such a solution c_i always exists, for which $\|\sum c_i v_i\|_\infty$ is bounded, for any initial settings of $r_1, \dots, r_N$, as long as they are all large enough.

We show that such a solution always exists, with $|c_i| \le 7$. In order to show it, we first fix some solution with the correct parity, and then correct it to a low discrepancy solution, by adding an even number of copies of each vector. In order to do that, we study the integer lattice $\mathcal{L}$ spanned by the vectors $v_1, \dots, v_N$, as our correction comes from $2\mathcal{L}$. We show that $\mathcal{L} = \{x \in \mathbb{Z}^m : \sum x_i = 0 \mod t\}$, which was already proved by Wilson [15] in a more general scenario. However, we need an additional property: vectors in $\mathcal{L}$ are efficiently spanned by $v_1, \ldots, v_N$. This allows us to perform the above correction efficiently, keeping the number of times that each v_i is repeated bounded. Putting that together, we obtain the result.

5.2 Proof of Theorem 13

Initially, we investigate the lattice spanned by the vectors $v_1, \ldots, v_N$. As the sum of the coordinates of each of them is t, they sit within the lattice

$$\mathcal{L} = \left\{x \in \mathbb{Z}^m : \sum x_i \equiv 0 \mod t\right\}.$$

We first show that they span this lattice, and moreover, they do so effectively.

▶ **Lemma 14.** *For any $w \in \mathcal{L}$ there exist $a_1, \ldots, a_N \in \mathbb{Z}$ such that $\sum a_i v_i = w$. Moreover, $|a_i| \le A$ for all $i \in [N]$ where $A = \frac{2\|w\|_1}{\binom{m-2}{t-1}} + 2$.*

Proof. Assume first that we have $\sum w_i = 0$. We will later show how to reduce to this case. Pair the positive and negative coordinates of w. For $L = \|w\|_1/2$ let $(i_1, j_1), \ldots, (i_L, j_L)$ be pairs of elements of $[N]$ such that: if (i, j) is a pair then $w_i > 0, w_j < 0$; each $i \in [m]$ with $w_i > 0$ appears w_i times as the first element in a pair; and each $j \in [m]$ with $w_j < 0$ appears $-w_j$ times as the second element in a pair. For any $\ell \in [L]$ choose $S_\ell \subset [m]$ of size $t-1$. Set $I_\ell = S_\ell \cup \{i_\ell\}$ and $J_\ell = S_\ell \cup \{j_\ell\}$. Identifying $[N]$ with subsets of $[m]$ of size t, we have

$$w = \sum_{\ell=1}^{L} v_{I_\ell} - v_{J_\ell}.$$

We choose the sets $S_1, \ldots, S_L$ to minimize the maximum number of times that each vector from $\{v_1, \ldots, v_N\}$ is repeated in the decomposition. When we choose S_ℓ, we can choose one of $M = \binom{m-2}{t-1}$ many choices. There is a choice for S_ℓ such that both I_ℓ and J_ℓ appeared thus far less than $2\ell/M$ times. Choosing such a set, we maintain the invariant that after choosing $S_1, \ldots, S_\ell$, each vector is repeated at most $2\ell/M + 1$ times. Thus, at the end each vector is repeated at most $2L/M + 1$ times.

In the general case, we have $\sum w_i = st$, where we may assume $s > 0$. We apply the previous argument to $w - (v_{i_1} + \ldots + v_{i_s})$, whose coordinates sum to zero. We choose $i_1, \ldots, i_s \in [N]$ (potentially with repetitions) so as to minimize the maximum number of times that each vector participates; this number is $\lceil s/N \rceil \le \|w\|_1/M + 1$. Combining the two estimates, we obtain that at the end each vector is repeated at most $4L/M + 2 = 2\|w\|_1/M + 2$ times. ◀

▶ **Lemma 15.** *For any $b_1, \ldots, b_N \in \{0, 1\}$ there exist $c_1, \ldots, c_N \in \mathbb{Z}$ such that*
(i) *$c_i \equiv b_i \mod 2$.*
(ii) *$\|\sum c_i v_i\|_\infty \le 2$.*
(iii) *$|c_i| \le 7$ for all $i \in [N]$.*

Proof. As a first step, choose $z_i \in \{-1, 0, 1\}$ such that $z_i = 0$ if $b_i = 0$, and $z_i \in \{-1, 1\}$ chosen uniformly if $b_i = 1$. Let $u = \sum z_i v_i$. Note that for $j \in [m]$, if there are k_j indices $i \in [N]$ for which $(v_i)_j = 1$ and $b_i = 1$, then $\mathbb{E}_z[u_j^2] = k_j$. Thus,

$$\mathbb{E}_z[\|u\|_2^2] = \sum k_j \le Nt.$$

Thus, with probability at least 1/2, $\|u\|_2 \le \sqrt{2Nt}$ and hence $\|u\|_1 \le \sqrt{2Ntm}$. Fix such a u.

Next, we choose $w \in \mathcal{L}$ such that $\|u - 2w\|_\infty$ is bounded. If we only wanted that $w \in \mathbb{Z}^m$ we could simply choose $q \in \{0,1\}^m$ with $q_i = u_i \mod 2$ and take $w = (u-q)/2$. In order to guarantee that $w \in \mathcal{L}$, namely that $\sum w_i = 0 \mod t$, we change at most t coordinates in q by adding or subtracting 2. Thus, we obtain $q \in \{-2,-1,0,1,2\}^m$ where $q_i \equiv u_i \mod 2$ and set $w = (u-q)/2 \in \mathcal{L}$. We have $\|u - 2w\|_\infty \le 2$.

Next, we apply Lemma 14 to w. We obtain a decomposition $w = \sum a_i v_i$. This implies that if we set $c_i = z_i - 2a_i$ then indeed $c_i \equiv b_i \mod 2$ and $\|\sum c_i v_i\|_\infty = \|u - 2w\|_\infty \le 2$. To bound $|c_i|$, note that $\|w\|_1 \le \|u\|_1/2 + m$. We have by Lemma 14 that $|a_i| \le A$ for

$$A = 2 + \eta \le 3,$$

where

$$\eta = 2\frac{\|w\|_1}{\binom{m-2}{t-1}} \le O\left(\frac{\sqrt{mt\binom{m}{t}}}{\binom{m-2}{t-1}}\right) \le O\left(\frac{m^{3/2}}{\binom{m}{t}^{1/2}}\right) \le 1,$$

whenever $4 \le t \le m-4$ and m is large enough, as is easily verified by the fact that the last term is a decreasing function of m. ◀

Proof of Theorem 13. Assume that $r_1, \ldots, r_N \ge 7$. By Lemma 15, there exists $c_i \in \mathbb{Z}$ such that $c_i \equiv r_i \mod 2$, $|c_i| \le 7$ and $\|\sum c_i v_i\|_\infty \le 2$. For each $i \in [N]$, we color $|c_i|$ of the vectors v_i with $\operatorname{sign}(c_i) \in \{-1,+1\}$ and the remaining $r_i - |c_i|$ vectors with alternating $+1$ and -1 colors (so that their contribution cancels, since $r_i - |c_i|$ is even). The total coloring produces exactly the vector $\sum c_i v_i$, which as guaranteed has discrepancy bounded by 2. ◀

Acknowledgments. The authors wish to thank Aravind Srinivasan for presenting this problem during a discussion at the IMA Workshop on the Power of Randomness in Computation.

References

1 Noga Alon and Joel H. Spencer. *The probabilistic method.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, New York, Chichester, Weinheim, 2000.

2 Wojciech Banaszczyk. Balancing vectors and Gaussian measures of n-dimensional convex bodies. *Random Structures & Algorithms*, 12(4):351–360, 1998.

3 Nikhil Bansal, Daniel Dadush, and Shashwat Garg. An algorithm for Komlós conjecture matching Banaszczyk's bound. *arXiv preprint arXiv:1605.02882*, 2016.

4 József Beck and Tibor Fiala. Integer-making theorems. *Discrete Applied Mathematics*, 3(1):1–8, 1981.

5 Debe Bednarchak and Martin Helm. A note on the Beck-Fiala theorem. *Combinatorica*, 17(1):147–149, 1997.

6 Boris Bukh. An improvement of the Beck-Fiala theorem. *arXiv preprint arXiv:1306.6081*, 2013.

7 Bernard Chazelle. *The discrepancy method: randomness and complexity.* Cambridge University Press, Cambridge, New York, 2000.

8 Paul Erdös and Rényi Alfréd. On a classical problem of probability theory. *Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei*, 6:215–220, 1961.

9 Paul Erdös and Laszlo Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and Finite Sets (to Paul Erdös on his 60th birthday)*, II:609–627, 1975.

10 Martin Helm. On the Beck-Fiala theorem. *Discrete mathematics*, 207(1):73–87, 1999.

11 Jiri Matoušek. *Geometric discrepancy: An illustrated guide*, volume 18. Springer Science & Business Media, 2009.

12 Robin A Moser. A constructive proof of the Lovász local lemma. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 343–350. ACM, 2009.

13 Robin A Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM (JACM)*, 57(2):11, 2010.

14 Joel Spencer. Six standard deviations suffice. *Transactions of the American Mathematical Society*, 289(2):679–706, 1985.

15 Richard M. Wilson. A diagonal form for the incidence matrices of t-subsets vs. k-subsets. *European Journal of Combinatorics*, 11(6):609–615, 1990.

The Niceness of Unique Sink Orientations

Bernd Gärtner[1] **and Antonis Thomas**[2]

1 **Department of Computer Science, Institute of Theoretical Computer Science, ETH Zürich, Switzerland**
gaertner@inf.ethz.ch

2 **Department of Computer Science, Institute of Theoretical Computer Science, ETH Zürich, Switzerland**
athomas@inf.ethz.ch

Abstract

Random Edge is the most natural randomized pivot rule for the simplex algorithm. Considerable progress has been made recently towards fully understanding its behavior. Back in 2001, Welzl introduced the concepts of *reachmaps* and *niceness* of Unique Sink Orientations (USO), in an effort to better understand the behavior of Random Edge. In this paper, we initiate the systematic study of these concepts. We settle the questions that were asked by Welzl about the niceness of (acyclic) USO. Niceness implies natural upper bounds for Random Edge and we provide evidence that these are tight or almost tight in many interesting cases. Moreover, we show that Random Edge is polynomial on at least $n^{\Omega(2^n)}$ many (possibly cyclic) USO. As a bonus, we describe a derandomization of Random Edge which achieves the same asymptotic upper bounds with respect to niceness.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics

Keywords and phrases random edge, unique sink orientation, random walk, reachmap, niceness

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.30

1 Introduction

One of the most prominent open questions in the theory of optimization is whether linear programs can be solved in strongly polynomial time. In particular, it is open whether there exists a pivot rule for the simplex method whose number of steps can be bounded by a polynomial function of the number of variables and constraints. For most deterministic pivot rules discussed in the literature, exponential lower bounds are known. The first such bound was established for Dantzig's rule by Klee and Minty in their seminal 1972 paper [20]; this triggered a number of similar results for many other rules; only in 2011, Friedmann solved a longstanding open problem by giving a superpolynomial lower bound for Zadeh's rule [8].

On the other hand, there exists a *randomized* pivot rule, called *Random Facet*, with an expected *subexponential* number of steps in the worst case. This bound was found independently by Kalai [18] as well as Matoušek, Sharir and Welzl [23] in 1992. Interestingly, the proofs employ only a small number of combinatorial properties of linear programs. As a consequence, the subexponential upper bound for the Random Facet pivot rule holds in a much more general abstract setting that encompasses many other (geometric) optimization problems for which strongly polynomial algorithms are still missing [23].

This result sparked a lot of interest in abstract optimization frameworks that generalize linear programming. The most studied such framework, over the last 15 years, is that of *unique sink orientations* (USO). First described by Stickney and Watson already in 1978 as abstract models for P-matrix linear complementarity problems (PLCPs) [28], USO were revived by Szabó and Welzl in 2001 [29]. Subsequently, their structural and algorithmic properties were studied extensively ([26],[27],[22],[12],[7],[2],[16],[13],[19],[17]). In a nutshell, a USO is an orientation of the n-dimensional hypercube graph, with the property that there is a unique sink in every subgraph induced by a nonempty face. The algorithmic problem associated to a USO is that of finding the unique global sink, in an oracle model that allows us to query any given vertex for the orientations of its incident edges.

In recent years, USO have in particular been looked at in connection with another randomized pivot rule, namely *Random Edge* (RE for short). This is arguably the most natural randomized pivot rule for the simplex method, and it has an obvious interpretation also on USO: at every vertex pick an edge uniformly at random from the set of outgoing edges and let the other endpoint of this edge be the next vertex. The path formed constitutes a *random walk*. Ever since the subexponential bound for Random Facet was proved in 1992, researchers have tried to understand the performance of Random Edge. This turned out to be very difficult, though. Unlike Random Facet, the Random Edge algorithm is non-recursive, and tools for a successful analysis were simply missing. A superexponential lower bound on cyclic USO was shown by Morris in 2002 [25], but there was still hope that Random Edge might be much faster on acyclic USO (AUSO).

Only in 2006, a superpolynomial and subexponential *lower* bound for Random Edge on AUSO was found by Matoušek and Szabó [24] and, very recently, pushed further by Hansen and Zwick [17]. While these are not lower bounds for actual linear programs, the results demonstrate the usefulness of the USO framework: it is now clear that the known combinatorial properties of linear programming are not enough to show that Random Edge is fast. Note that, in 2011, Friedmann, Hansen and Zwick proved a subexponential lower bound for Random Edge on actual linear programs, "killing" yet another candidate for a polynomial-time pivot rule [9].

Still, the question remains open whether Random Edge also has a subexponential *upper* bound. As there already is a subexponential algorithm, a positive answer would not be an algorithmic breakthrough; however, as Random Edge is notoriously difficult to analyze, it might be a breakthrough in terms of novel techniques for analyzing this and other randomized algorithms. The currently best upper bound on AUSO is an exponential improvement over the previous (almost trivial) upper bounds, but the bound is still exponential, 1.8^n [16].

In this paper, we initiate the systematic study of concepts that are tailored to Random Edge on USO (not necessarily only AUSO). These concepts – *reachmaps* and *niceness* of USO – were introduced by Welzl [30], in a 2001 workshop as an interesting research direction. At that time, it seemed more promising to work on algorithms other than Random Edge; hence, this research direction remained unexplored and the problems posed by Welzl remained open. Now that the understanding of Random Edge on USO has advanced a lot we hope that these "old" concepts will finally prove useful, probably in connection with other techniques.

The reachmap of a vertex is the set of all the coordinates it can reach with a directed path, and a USO is i-nice if for every vertex there is a directed path of length at most i to another vertex with smaller reachmap. Welzl pointed out that the concept of niceness provides a natural upper bound for the Random Edge algorithm. Furthermore, he asks the following question: "Clearly every unique sink orientation of dimension n is n-nice. Can we do better? In particular what is the general niceness of acyclic unique sink orientations?"

We settle these questions, in Section 4, by proving that for AUSO $(n-2)$-nice is tight, meaning that $(n-2)$ is an upper bound on the niceness of all AUSO and there are AUSO that are *not* $(n-3)$-nice. For cyclic USO we argue that n-nice is tight. In Section 2, we give the relevant definitions and in Section 3 we show an upper bound of $O(n^{i+1})$ for the number of steps RE takes on an i-nice USO. In addition, we describe a derandomization of RE which also takes at most $O(n^{i+1})$ on an i-nice USO, thus matching the behavior of RE.

Finally, we include two brief notes in Section 3. The first argues that RE needs at most a quadratic number of steps in at least $n^{\Theta(2^n)}$ many, possibly cyclic, USO. The second that RE can solve the AUSO instances that have been designed as lower bounds for other algorithms (e.g. Random Facet [21],[10] or Bottom Antipodal [27]) in polynomial time. All the necessary details for these two notes will be provided in the full version [14].

2 Preliminaries

We use the notation $[n] = \{1, \dots n\}$. Let $Q^n = 2^{[n]}$ be the set of vertices of the n-dimensional hypercube. A vertex of the hypercube $v \in Q^n$ is denoted by the set of coordinates it contains. The symmetric difference of two vertices, denoted as $v \oplus u$ is the set of coordinates in which they differ. Now, let $J \in 2^{[n]}$ and $v \in Q^n$. A *face* of the hypercube, $F_{J,v}$, is defined as the set of vertices that are reached from v over the coordinates defined by any subset of J, i.e. $F_{J,v} = \{u \in Q^n | v \oplus u \subseteq J\}$. The dimension of the face is $|J|$. We call edges the faces of dimension 1, e.g. $F_{\{j\},v}$, and vertices the faces of dimension 0. The faces of dimension $n-1$ are called facets. For $k \leq n$ we call a face of dimension k a k-face.

Let $v, u \in Q^n$. By $|v \oplus u|$ we denote the Hamming distance (size of the symmetric difference) of v and u. Given $v \in Q^n$, we define the neighborhood of v as $\mathcal{N}(v) = \{u \in Q^n | \; |v \oplus u| = 1\}$. Now, let ψ be an orientation of the edges of the n-dimensional hypercube. Let $v, u \in Q^n$. The notation $v \xrightarrow{j} u$ (w.r.t ψ) means that $F_{\{j\},v} = \{v, u\}$ and that the corresponding edge is oriented from v to u in ψ. Sometimes we write $v \to u$, when when the coordinate is irrelevant. An edge $v \xrightarrow{j} u$ is forward if $j \in u$ and otherwise we say it is backward.

We say that ψ is a *Unique Sink Orientation* (USO) if every non-empty face has a unique sink. In the rest we write n-USO to mean a USO over Q^n. Here n is always used to mean the dimension of the corresponding USO. Consider a USO ψ; we define its outmap s_ψ, in the spirit of Szabó and Welzl [29]. The *outmap* is a function $s_\psi : Q^n \to 2^{[n]}$, defined by $s_\psi(v) = \{j \in [n] | v \xrightarrow{j} v \oplus \{j\}\}$ for every $v \in Q^n$. A sink of a face $F_{J,v}$ is a vertex $u \in F_{J,v}$, such that $s_\psi(u) \cap J = \emptyset$. We mention the following lemma w.r.t. the outmap function.

▶ **Lemma 1** ([29]). *For every USO ψ, s_ψ is a bijection.*

The algorithmic problem for a USO ψ is to find the global sink, i.e. find $t \in Q^n$ such that $s_\psi(t) = \emptyset$. The computations take place in the *vertex oracle* model: We have an oracle that given a vertex $v \in Q^n$, returns $s_\psi(v)$ (vertex evaluation). This is the standard computational model in the USO literature and all the upper and lower bounds refer to it.

Reachmap and niceness. We are now ready to define the central concepts of this paper. Given vertices $v, u \in Q^n$ we write $v \rightsquigarrow u$ if there exists a directed path from v to u (in ψ). We use $d(v, u)$ to denote the length of the shortest directed path from v to u; if there is no such path then we have $d(v, u) = \infty$ and otherwise we have $d(v, u) \geq |v \oplus u|$. The following lemma is well-known and easy to prove by induction on $|v \oplus u|$.

▶ **Lemma 2.** *For every USO ψ, let $F \subseteq Q^n$ be a face and u the sink of this face. Then, for every vertex $v \in F$ we have $d(v, u) = |v \oplus u|$.*

Subsequently, we define the *reachmap* $r_\psi : Q^n \to 2^{[n]}$, for every $v \in Q^n$, as:

$$r_\psi(v) = s_\psi(v) \cup \{j \in [n] | \exists u \in Q^n \text{ s.t. } v \rightsquigarrow u \text{ and } j \in s_\psi(u)\}.$$

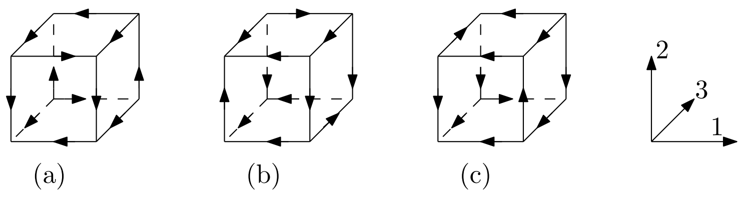


Figure 1 Examples of 3-dimensional USO: (a) Klee-Minty, which is 1-nice. (b) The only 2-nice 3-dimensional AUSO which is not 1-nice. (c) The only cyclic USO in 3 dimensions, which is 3-nice.

Intuitively, the reachmap of a vertex contains all the coordinates that the vertex can reach with a directed path. We say that vertex $v \in Q^n$ is i-covered by vertex $u \in Q^n$, if $d(v,u) \leq i$ and $r_\psi(u) \subset r_\psi(v)$ (proper inclusion). Then, we say that a USO ψ is *i-nice* if every vertex $v \in Q^n$ (except the global sink) is i-covered by some vertex $u \in Q^n$. Of course, every n-USO ψ is n-nice since every vertex v is n-covered by the sink t. Moreover, $r_\psi(v) \supseteq v \oplus t$, for every vertex $v \in Q^n$.

It is not difficult to observe that every USO in 1 or 2 dimensions is 1-nice, but the situation changes in 3 dimensions. Consider the illustration in Figure 1.

Let us note that the AUSO in Figure 1(b) is the largest AUSO which is not $(n-2)$-nice. As we prove in Theorem 8, every n-AUSO with $n \geq 4$ is $(n-2)$-nice.

Algorithmic properties of the reachmap. Our focus lies mostly on the concept of niceness. Nevertheless, we briefly discuss some of the algorithmic properties of the reachmap here.

It was proved by the authors, in [13], that when given an AUSO ψ described succinctly by a Boolean circuit, and two vertices s and t, deciding if $s \rightsquigarrow t$ is $PSPACE$-complete. More recently, Fearnley and Savani [6] proved that deciding whether the Bottom Antipodal algorithm (this is the algorithm that from a vertex v jumps to vertex $v \oplus s_\psi(v)$), started at vertex v will ever encounter a vertex v' such that $j \in s_\psi(v')$, for a given coordinate j, is $PSPACE$-complete. This line of work was initiated in [1] and further developed in [4] and [5] and aims at understanding the computational power of pivot algorithms [6]. Below, we provide a related theorem: it is $PSPACE$-complete to decide if a coordinate is in the reachmap of a given vertex in an AUSO. It is, thus, computationally hard to discover the reachmap of a vertex.

▶ **Theorem 3.** *Let ψ be an n-AUSO (described succinctly by a Boolean circuit), $v \in Q^n$ and $j \in [n]$. It is $PSPACE$-complete to decide whether $j \in r_\psi(v)$.*

The theorem follows from the results of [13] that we mention above. A proof is included in the full version [14]. Finally, we want to note that it is natural to upper bound algorithms on AUSO by the reachmap of the starting vertex. Any reasonable path-following algorithm that solves an AUSO ψ in c^n steps, for some constant c, can be bounded by $c^{|r_\psi(s)|}$ where s is the starting vertex. The reason is that the algorithm will be contained in the cube $F_{r_\psi(s),s}$ of dimension $|r_\psi(s)|$. Moreover, we claim that this is also possible for algorithms that are not path-following. As an example we give in the full version of this paper [14] a variant of the Fibonacci Seesaw algorithm of [29] that runs in time $c^{|r_\psi(s)|}$ for some $c < \phi$ (the golden ratio).

3 Random Edge on i-nice USO

In this section we describe how RE behaves on i-nice USO. We give a natural upper bound and argue that it is tight or almost tight in many situations. In addition, we give a simple derandomization of RE, which asymptotically achieves the same upper bound. Firstly, we consider the following natural upper bound.

▶ **Theorem 4.** *Started at any vertex of an i-nice USO, Random Edge will perform an expected number of at most $O(n^{i+1})$ steps.*

Proof. For every vertex v, there is a directed path of length at most i to a *target* $t(v)$, some fixed vertex of smaller reachmap. At every step, we either reduce the distance to the current target (if we happen to choose the right edge), or we start over with a new vertex and a new target. The expected time it takes to reach *some* target vertex can be bounded by the expected time to reach state 0 in the following Markov chain with states $0, 1, \ldots, i$ (representing distance to the current target): at state $k > 0$, advance to state $k - 1$ with probability $1/n$, and fall back to state i with probability $(n - 1)/n$. A simple inductive proof shows that state 0 is reached after an expected number of $\sum_{k=1}^{i} n^k = O(n^i)$ steps. Hence, after this expected number of steps, we reduce the reachmap size, and as we have to do this at most n times, the bound follows. ◀

Already, we can give some first evidence on the usefulness of niceness for analyzing RE: Decomposable orientations have been studied extensively in literature. The fact that RE terminates in $O(n^2)$ steps on them has been known at least since the work of Williamson-Hoke [31]. Let a coordinate be *combed* if all edges on this coordinate are directed the same way. Then, a cube orientation is *decomposable* if in every face of the cube there is a combed coordinate. The class of decomposable orientations, known to be AUSO, contains the Klee-Minty cube [20] (defined combinatorially in [27]). It is straightforward to argue that such orientations are 1-nice and, thus, our upper bound from Theorem 4 is also quadratic. Moreover, quadratic lower bounds have been proved for the behavior of RE on Klee-Minty cubes [3]. We conclude that, for 1-nice USO, the upper bound in Theorem 4 is optimal.

Counting 1-nice. We have mentioned that the class of decomposable USO are 1-nice. This class is the previously known largest class of USO, where Random Edge is polynomial. The number of decomposable USO is $2^{\Theta(2^n)}$ (a proof for this is included in the full version [14]). We can now argue that the class of 1-nice USO is much larger than the class of decomposable ones, and also contains cyclic USO. To achieve the lower bound we use the same technique that Matoušek [22] used to give a lower bound on the number of all USO. The upper bound is proved also in [22]. Thus, we have the number of 1-nice USO is asymptotically (in the exponent) the same as of all USO.

▶ **Theorem 5.** *The number of 1-nice n-dimensional USO is $n^{\Theta(2^n)}$.*

Proof of Theorem 5. Consider the following inductive construction. Let A_1 be any 1-dimensional USO. Then, we construct A_2 by taking any 1-dimensional USO A_1' and directing all edges on coordinate 2 towards A_1. In general, to construct A_{k+1}: we take A_k and put antipodally any k-dimensional USO A_k'. Then, we direct all edges on coordinate $(k + 1)$ towards A_k. This is safe by the Product Lemma (this is one of the two main USO constructing lemmas from [26]). This construction satisfies the following property: for every vertex, the minimal face that contains this vertex and the global sink has a combed coordinate. We call such a USO "target-combed". It constitutes a generalization of decomposable USO. An illustration appears in Figure 2.

The construction is 1-nice since for every vertex (except the sink) there is an outgoing coordinate that can never be reached again. At every iteration step from k to $k + 1$ we can

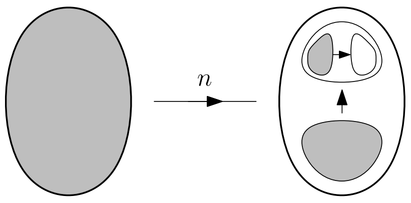


Figure 2 A target-combed n-USO. The two larger ellipsoids represent the two antipodal facets A_{n-1} and A'_{n-1} and, similarly, for the smaller ones. The combed coordinates are highlighted. The gray subcubes can be oriented by any USO.

embed, in one of the two antipodal k-faces, any USO. Thus, we can use the lower bounds of [22], that give us a $\left(\frac{k}{e}\right)^{2^{k-1}}$ (assuming $k \geq 2$) lower bound for a k-face. Summing up, we get:

$$uso1nice(n) \geq \sum_{k=1}^{n-1} uso(k) > uso(n-1) = \left(\frac{n-1}{e}\right)^{2^{n-2}}$$

where $uso1nice(n)$ and $uso(n)$ is the number of n-dimensional 1-nice USO and general USO respectively. Thus, $uso1nice(n) = n^{\Omega(2^n)}$. The upper bound in the statement of the theorem is from the upper bound on the number of all USO, by Matoušek [22]. ◀

The niceness of known lower bound constructions

As further motivation for the study of niceness of USO, we want to mention that RE can solve the AUSO instances that were designed as lower bounds for other algorithms in polynomial time. This is because of provable upper bounds on the niceness of those constructions. With similar arguments, upper bounds on the niceness of the AUSO that serve as subexponential lower bounds for RE can be shown; thus, RE has upper bounds on these constructions that are almost matching to the lower bounds. The upper bound for RE on the cyclic USO of Morris [25] is asymptotically matching the lower bound. We summarize the relevant information in the following table and describe the details on how to obtain it in the full version [14].

Algorithm	Reference	Lower bound	Niceness	RE Upper bound
Random Facet	[21],[10]	$2^{\Theta(\sqrt{n})}$	1	$O(n^2)$
Bottom Antipodal	[27]	$\Omega(\sqrt{2}^n)$	2	$O(n^3)$
RE acyclic	[24]	$2^{\Omega(n^{1/3})}$	$n^{1/3}$	$2^{O(n^{1/3}\log n)}$
RE acyclic	[17]	$2^{\Omega(\sqrt{n\log n})}$	$\sqrt{n}$	$2^{O(\sqrt{n}\log n)}$
RE cyclic	[25]	$\frac{n-1}{2}!$	n	$n^{O(n)}$

A derandomization of Random Edge

Consider the *join* operation. Given two vertices u, v, $join(u, v)$ is a vertex w such that $u \rightsquigarrow w$ and $v \rightsquigarrow w$. We can compute $join(u, v)$ as follows: by Lemma 1, there must be a coordinate, say j, such that $j \in s_\psi(u) \oplus s_\psi(v)$. Assume, w.l.o.g., that $j \in s_\psi(u)$. Consider the neighbor u' of u such that $u \xrightarrow{j} u'$. Recursively compute $join(u', v)$. It can be seen by induction on $|u \oplus v|$ that the *join* operation takes $O(n)$ time. Similarly, we talk about a join of a set S of vertices. A $join(S)$ is a vertex w such that ever vertex in S has a path to it. We can compute $join(S)$ by iteratively joining all the vertices in S.

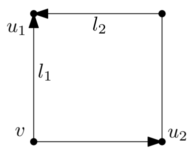


Figure 3 We have $u, w \in AV$, $l \in AC$, $l \notin s_\psi(u)$ and $\{l\} \neq (v \oplus u)$. Thus, the edge $F_{j,w}$ has to be outgoing for w. Hence, $w \rightsquigarrow u$ and the algorithm removes w from AV and l from AC.

Furthermore, let $\mathcal{N}^+(v) = \{u \in \mathcal{N}(v) | v \to u\}$ denote the set of out-neighbors of a vertex v. In the subsequent lemma, we argue that the vertices in $\mathcal{N}^+(v)$ can be joined with linearly many vertex evaluations.

▶ **Lemma 6.** *Let ψ be an n-USO and $v \in Q^n$ a vertex. There is an algorithm that joins the vertices in $\mathcal{N}^+(v)$ with $|s_\psi(v)|$ many vertex evaluations.*

Proof. First, we evaluate all the vertices in $\mathcal{N}^+(v)$. We maintain a set of active vertices AV and a set of active coordinates AC. Initialize $AV = \mathcal{N}^+(v)$ and $AC = s_\psi(v)$. The algorithm keeps the following invariants: every vertex that gets removed from AV has a path to some vertex in AV; also for every vertex u s.t. $v \xrightarrow{l} u$, $u \in AV$ if and only if $l \in AC$.

Then, for each $u \in AV$: for each $l \in AC$: if $l \notin s_\psi(u)$ and $\{l\} \neq (u \oplus v)$ then we update $AC \leftarrow AC \setminus \{l\}$ and $AV \leftarrow AV \setminus (v \oplus \{l\})$. See Figure 3.

If in the above loop the vertex u is the sink of the face $F_{AC,u}$ then terminate and return $v' = u$. Of course, in this case every vertex in AV has a path to u. Otherwise the loop will terminate when there is no coordinate in AC that satisfies the conditions above. In this case we have that $\forall u \in AV$, u is the source of the face $F_{AC \setminus (u \oplus v), u}$. That is, it is the source of the face spanned by the vertex and all the active coordinates AC except the one that connects it to v. In this case, we return the vertex $v' = (v \oplus AC)$. We have that every vertex in AV has a path to v': this is because in any USO the source has a path to every vertex (this can be proved similarly to Lemma 2). ◀

Using Lemma 6, we can now argue that there exists a derandomization of Random Edge that asymptotically matches the upper bound of Theorem 4.

▶ **Theorem 7.** *There is a deterministic algorithm that finds the sink of an i-nice n-USO ψ with $O(n^{i+1})$ vertex evaluations.*

Proof. Let v be the current vertex. Consider the set $R_i \subseteq 2^{[n]}$ of vertices that are reachable along directed paths of length at most i from v. Since ψ is i-nice, we know that at least one of them has strictly smaller reachmap. In particular, any vertex reachable from all the vertices in R_i has a smaller reachmap. Thus, we compute a *join* of all the vertices in R_i.

Consider the set R_{i-1}. The size of R_{i-1} is bounded by $|R_{i-1}| \leq \sum_{k=0}^{i-1} \binom{n}{k} \leq \sum_{k=0}^{i-1} n^k$ and, thus, $|R_{i-1}| = O(n^{i-1})$. Every vertex in R_i can be reached in one step from some vertex in R_{i-1}. Assume that none of the vertices in R_{i-1} is the sink; otherwise, the algorithm is finished. Then, for every vertex $v \in R_{i-1}$ we join $\mathcal{N}^+(v)$ with the algorithm from Lemma 6, with $O(n)$ vertex evaluations. Therefore, with $O(n^i)$ vertex evaluations we have a set S of $O(n^{i-1})$ many vertices and each $v' \in S$ is a join of $\mathcal{N}^+(v)$ for some vertex $v \in R_{i-1}$.

The next step is to join all the vertices in set S, using the algorithm at the beginning of the current section, which takes $O(n)$ for each pair of vertices. Hence, the whole procedure will take an additional $O(n^i)$ vertex evaluations. The result is a vertex u that joins all the vertices in R_i and thus i-covers v. Because the size of the reachmap decreases by at least one in each round, we conclude that this algorithm will take at most $O(n^{i+1})$ steps.

Finally, note that to achieve this upper bound we do not need to know that the input USO is i-nice. Instead, we can iterate through the different values of $i = 1, 2, \dots$ without changing the asymptotic behavior of the algorithm. ◀

4 Bounds on niceness

In this section we answer the questions originally posed in [30] by providing matching upper and lower bounds on the niceness of USO and AUSO.

For cyclic USO, the cubes designed by Morris as a lower bound for the behavior of RE [25] are n-nice but not $(n-1)$-nice and, hence, match the trivial upper bound. Here, we sketch a construction that is n-nice (but not $(n-1)$-nice) and we give an explicit description in the full version [14].

The idea for such a construction is quite simple, intuitively. Let ψ be a cyclic n-USO over Q^n that contains a directed cycle such that the edges that participate span all the coordinates. Then, every vertex v on the cycle has $r_\psi(v) = [n]$. Now consider the sink t and assume the n vertices in $\mathcal{N}(t)$ participate in the cycle. By Lemma 2, every vertex has a path to t. This path has to go through one of the vertices in $\mathcal{N}(t)$. It follows that every $v \in Q^n \setminus \{t\}$ has $r_\psi(v) = [n]$. Therefore, the vertex antipodal from t is only n-covered (by t).

The Morris cyclic USO satisfies the properties described above and, thus, it cannot be $(n-1)$-nice. An example in 3 dimensions appears in Figure 1; this USO satisfies the properties we explain above. The construction we describe in the full version [14] is much simpler than the Morris cube; it is an n-USO that contains a simple cycle over $2n$ vertices, n of which are the vertices in $\mathcal{N}(t)$. Half of the edges that participate on the cycle are backward and every other edge in the USO is forward. For the rest of this section, we will turn our attention to AUSO.

4.1 An upper bound for AUSO

We prove an upper bound on the niceness of AUSO which, as we will see in the next section, is tight. We utilize the concept of Completely Unimodal Numberings (CUN), which was studied by Williamson-Hoke [31] and Hammer *et al.* [15]. To the best of our knowledge, this is the first time CUN is used to prove structural results for AUSO. A CUN on the hypercube Q^n means that there is a bijective function $\phi : Q^n \to \{0, \dots, 2^n - 1\}$ such that in every face F there is exactly one vertex v such that $\phi(v) < \phi(u)$, for every $u \in \mathcal{N}(v) \cap F$. It is known, e.g. from [31], that for every AUSO there is a corresponding CUN, which can be constructed by topologically sorting the AUSO.

In the proof of the theorem below we will use the following notation: w^k is the vertex that has $\phi(w^k) = k$, w.r.t. some fixed CUN ϕ. An easy, but crucial observation concerns the three lowest-numbered vertices w^0, w^1, w^2. Of course, $w^1 \to w^0$ (where w^0 is the global sink); otherwise, w^1 would have been a second global minimum. Moreover, $w^2 \to w^j$ for exactly one $j \in \{0, 1\}$. It follows, that both w^1 and w^2 are facet sinks. We are ready to state and prove the following theorem.

▶ **Theorem 8.** *Any n-AUSO, with $n \geq 4$, is $(n-2)$-nice.*

Consider the vertices w^0 and w^1 and let e be the edge that connects them. Let $w \in e$ be the unique out-neighbor of w^2 and w' the other vertex in e. W.l.o.g. assume $w = \emptyset, w' = \{1\}$ and $w^2 = \{2\}$. The situation can be depicted as:

$$\begin{tikzcd}
& w = \emptyset & \\
w^2 = \{2\} \arrow[ur] & & w' = \{1\} \arrow[ul, dash]
\end{tikzcd}$$

These three vertices have no outgoing edges to other vertices. Their outmaps and reachmaps are summarized in the table below.

vertex	outmap	reachmap	is sink of the facet
$w = \emptyset$	$\subseteq \{1\}$	$\subseteq \{1\}$	$F_{[n]\setminus\{1\},w}$
$w' = \{1\}$	$\subseteq \{1\}$	$\subseteq \{1\}$	$F_{[n]\setminus\{1\},w'}$
$w^2 = \{2\}$	$= \{2\}$	$\subseteq \{1, 2\}$	$F_{[n]\setminus\{2\},w^2}$

More precisely, the reachmap of w^2 is $\{2\}$ if $w = w^0$, and it is $\{1, 2\}$ if $w = w^1$.

▶ **Lemma 9.** *With w, w' as above, let $v \in Q^n \setminus \{w^0, [n]\}$. Then v is $(n-2)$-covered by some vertex in $\{w, w', w^2\}$.*

Proof. Vertex w^1 is covered by w^0 and w^2 by w^0 or w^1, so assume that v is some other vertex.

If v neither contains 1 nor 2, then v is in the facet $F_{[n]\setminus\{1\},w}$. Hence, $d(v, w) = |v \oplus w| \leq n-2$. This is because $F_{[n]\setminus\{1\},w}$ is $(n-1)$-dimensional and $2 \notin v$. Any coordinate that is part of the corresponding path is in the reachmap of v but not of w (whose reachmap is a subset of $\{1\}$). Hence, v is $(n-2)$-covered by w.

If v contains 1, then v is in the facet $F_{[n]\setminus\{1\},w'}$, and $|v \oplus w'| \leq n-2$ since $v \neq [n]$. As before, this implies that v is $(n-2)$-covered by the sink w' of the facet in question.

Finally, if v contains 2 but not 1, then v is in the face $F_{[n]\setminus\{1,2\},w^2}$, and $d(v, w^2) \leq n-2$. Again, any coordinate on a directed path from v to w^2 within this face proves that v is $(n-2)$-covered by the sink w^2 of the face. ◀

It remains to $(n-2)$-cover the vertex $v = [n]$. Let $m > 2$ be the smallest index such that w^m is not a neighbor of w, and assume w.l.o.g. that $w^k = \{k\}, 3 \leq k < m$. We have $w^k \to w$ for all these k by the vertex ordering. Furthermore, all other edges incident to w^k are incoming. We conclude that each $w^k, 3 \leq k < m$ has outmap equal to $\{k\}$ and, hence, is a facet sink. The reachmap of each such w^k is $\subseteq \{1, k\}$. The situation is depicted as:

$$\begin{tikzcd}
& & w = \emptyset & \\
w^{m-1} = \{m-1\} \arrow[urr] & \dots & w^2 = \{2\} \arrow[u] & w' = \{1\} \arrow[ul, dash] \\
& w^m = \{k, j\} \arrow[ul] \arrow[u] & &
\end{tikzcd}$$

Since w^m has at least one out-neighbor in $\{w', w^2, \ldots, w^{m-1}\}$, we know that $w^m = \{k, j\}$ for some $k < j \in [n]$. Moreover, the vertex ordering again implies that the outgoing edges of w^m are exactly the ones to its (at most two) neighbors among $w', w^2, \ldots, w^{m-1}$. Taking their reachmaps into account, we conclude that the reachmap of w^m is $\subseteq \{k, j, 1\}$.

▶ **Lemma 10.** *With w^m as above and $n \geq 4$, $v = [n]$ is $(n-2)$-covered by w^m.*

Proof. We first observe that w^m is the sink of the face $F_{[n]\setminus\{k,j\},w^m}$, since its outmap is $\subseteq \{k, j\}$. Vertex $v = [n]$ is contained in this $(n-2)$-face, hence there exists a directed path of length $d(v, w^m) = n-2$ from v to w^m in this face. Since $n \geq 4$, the path spans at least two coordinates and thus at least one of them is different from 1. This coordinate proves that v is $(n-2)$-covered by w^m. ◀

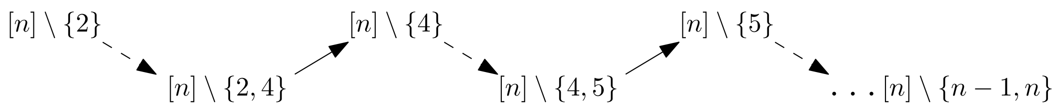


Figure 4 An illustration of the path starting at v'. The dashed edges are flipped backwards.

To sum up, we have now proved that every n-AUSO, with $n \geq 4$, is $(n-2)$-nice. All AUSO in one or two dimensions are 1-nice and the AUSO in three dimensions can be up to 2-nice (Figure 1b). This concludes the upper bounds on the niceness of AUSO.

4.2 A matching lower bound for AUSO

We prove a lower bound on the niceness of acyclic USO that matches the upper bound of Theorem 8. It follows (Corollary 6, [26]) from the Hypersink Reorientation Lemma [26] that in a USO we can *flip* any edge if the outmaps of the two vertices incident to it are the same (except the connecting coordinate). This gives rise to a particular family of USO, the Flip-Matching Orientations (FMO): those arise when we start with a uniform orientation, e.g. all edges are forward, and we flip the edges of an arbitrary matching. FMO have been studied in [26] and [24].

▶ **Theorem 11.** *There exists an n-AUSO ψ which is not i-nice, for $i < n-2$.*

Proof. Let $\psi_{\mathbb{U}}$ be the forward uniform orientation, i.e. the orientation where all edges are forward. We explain how to construct ψ, our target orientation starting from $\psi_{\mathbb{U}}$. With Q_k^n we denote the set of vertices that contain k coordinates, i.e. $|Q_k^n| = \binom{n}{k}$. We assume $n \geq 4$. The idea here is to construct an AUSO that has its source at $\emptyset$ and has the property that every vertex in $\bigcup_{i=0}^{n-3} Q_i^n$ has a full-dimensional reachmap.

Pick $v \in Q_{n-3}^n$ and assume w.l.o.g. that $v = [n] \setminus \{1,2,3\}$. Consider the 2-dimensional face $F_{\{1,2\},v}$ and direct the edges in this face backwards. This is the first step of the construction and it results in $s_\psi(v) = \{3\}$.

For the second step, consider the vertex $v' = [n] \setminus \{2\}$. We will flip $n-3$ edges in order to create a path starting at v'. First, we flip edge $F_{\{4\},[n]\setminus\{2\}}$. Then, for all $k \in \{4, \ldots, n-1\}$ we flip the edge $F_{\{k+1\},[n]\setminus\{k\}}$. This creates the path depicted in Figure 4.

Let U_3 be the set of vertices $U_3 = \{u \in Q_{n-3}^n | 3 \in u\}$. That is all the vertices of Q_{n-3}^n that contain the 3rd coordinate. For every $u \in U_3$ we flip the edge $F_{\{3\},u}$ (that is the edge incident to u on the 3rd coordinate). This is the third and last step of the construction of ψ.

▶ **Claim 12.** *ψ is a USO.*

The first step of the construction is to flip the four edges in $F_{\{1,2\},v}$. This is safe by considering that we first flip the two edges on coordinate 1; then, it is also safe to flip the two edges on coordinate 2. All the edges reversed at the second step of the construction (Figure 4) are between vertices in Q_{n-1}^n and Q_{n-2}^n, and, in addition those vertices are not neighbors to each other. Furthermore, all the edges reversed at the third step of the construction are on coordinate 3 and between vertices in Q_{n-3}^n and Q_{n-4}^n. Thus, all these edge flips are safe. Note however that edge flips do not neccassarily maintain acyclicity (e.g. the cyclic USO in Figure 1c is an FMO); we have to verify acyclicity in a different way.

▶ **Claim 13.** *There is no cycle in ψ.*

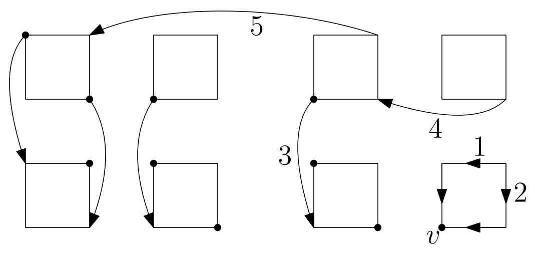


Figure 5 An example construction in 5 dimensions. Only the backward edges are noted. Each coordinate is labeled over a backward edge. The 5-dimensional cube is broken in 2-faces of coordinates 1,2. All the vertices in Q^n_{n-3} are noted with dots. Also, v is explicitly noted.

Clearly, a cycle has at least one backward and one forward edge in every coordinate it contains. Thus, there cannot be a cycle that involves coordinate 3 because no backward edge on a different coordinate, has a path connecting it to a backward edge on coordinate 3.

Consider the facet $F_{[n]\setminus\{3\},[n]}$ and the USO ψ', resulting from restricting ψ to the aforementioned facet. We can notice that ψ' is an FMO and the only backward edges are the ones attached to the path illustrated in Figure 4. Thus, a cycle has to use a part of this path. However, this path cannot be part of any cycles: a vertex on the higher level (vertices in Q^n_{n-1}) of the path has only two outgoing edges; one to the sink $[n]$ and one to the next vertex on the path. A vertex on the lower level Q^n_{n-2} has only one outgoing edge to the next vertex on the path. Also, the last vertex of the path $[n] \setminus \{n-1, n\}$ has only one outgoing edge to $[n] \setminus \{n\}$ which has only one outgoing edge to the sink $[n]$.

The fact that the facet $F_{[n]\setminus\{3\},v}$ has no cycle follows from the observation that there are backwards edges only on two coordinates which is not enough for the creation of a cycle (remember that in a USO a cycle needs to span at least three coordinates). This concludes the proof of Claim 13, which, combined with Claim 12, results in ψ being an AUSO.

▶ **Claim 14.** *Every vertex in $\bigcup_{i=0}^{n-3} Q^n_i$ has a full-dimensional reachmap.*

Firstly, we argue that v has $r_\psi(v) = [n]$. We have $s_\psi(v) = \{3\} \subset r_\psi(v)$. Then, $v \xrightarrow{3} u = [n] \setminus \{1,2\}$ and u has $s_\psi(u) = \{1,2\} \subset r_\psi(v)$. Vertex u is such that $u \xrightarrow{1} v' = [n] \setminus \{2\}$; v' is the beginning of the path described in Figure 4. The backwards edges on this path span every coordinate in $\{4, \dots, n\}$. This implies that $r_\psi(v') = \{2, 4, \dots, n\}$ and, since there is a path from v to v', $r_\psi(v') \subseteq r_\psi(v)$. Combined with the above, we have that $r_\psi(v) = [n]$.

Secondly, we argue that $\forall u \in Q^n_{n-3}$, $r_\psi(u) = [n]$. Vertex v is the sink of the facet $F_{[n]\setminus\{3\},v}$. It follows that every vertex in $Q^n_{n-3} \cap F_{[n]\setminus\{3\},v}$ has a path to v and thus has full dimensional reachmap. The vertices in U_3 (defined earlier), which are the rest of the vertices in Q^n_{n-3}, have backward edges on coordinate 3 and thus have paths to $F_{[n]\setminus\{3\},v}$. It follows that vertices in U_3 also have full dimensional reachmaps.

Any vertex in $\bigcup_{i=0}^{n-4} Q^n_i$ has a path to a vertex in Q^n_{n-3} since there are outgoing forward edges incident to any vertex in ψ (except the global sink at $[n]$). Thus, we have that $\forall u \in \bigcup_{i=0}^{n-3} Q^n_i$, $r_\psi(u) = [n]$ which proves the claim.

Finally, we combine the three Claims to conclude that the lowest vertex $\emptyset$ can only be covered by a vertex in Q^n_{n-2}. Therefore, ψ is *not* i-nice for any $i < n-2$, which proves the theorem. We include an example construction, for five dimensions, in Figure 5. ◀

5 Conclusions

In this paper we study the reachmaps and niceness of USO, concepts introduced by Welzl [30] in 2001. The questions that Welzl originally posed are now answered and the concepts explored further. We believe that these tools, or related ones, will prove useful in finally closing the gap between the lower and upper bounds known for RE. This will happen with either exponential lower bounds or with subexponential upper bounds. It is worth mentioning that these concepts are not only relevant for USO, but could also be defined on generalizations of USO, such as Grid USO [11] or Unimodal Numberings [15].

The authors of [17] define the concept of a (k, ℓ)-layered AUSO and use it to argue that their lower bounds are optimal under the method they use. Their concept is a generalization of niceness (on AUSO) but the exact relationship remains to be discovered. They pose the following questions: Are there AUSO that are not $(2^{O(\sqrt{n \log n})}, O(\sqrt{n/\log n}))$-layered? Are there small constants c, d such that all AUSO are $(c^n, dn/\log n)$-layered? We believe that the techniques of our proofs from Theorems 8 and 11 may be fruitful for answering these questions.

Acknowledgements. We would like to thank Thomas Dueholm Hansen and Uri Zwick for sharing their work [17] with us.

References

1 Ilan Adler, Christos H. Papadimitriou, and Aviad Rubinstein. On simplex pivoting rules and complexity theory. In Jon Lee and Jens Vygen, editors, *Integer Programming and Combinatorial Optimization – 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings*, volume 8494 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2014. doi:10.1007/978-3-319-07557-0_2.

2 Yoshikazu Aoshima, David Avis, Theresa Deering, Yoshitake Matsumoto, and Sonoko Moriyama. On the existence of Hamiltonian paths for history based pivot rules on acyclic unique sink orientations of hypercubes. *Discrete Applied Mathematics*, 160(15):2104–2115, 2012. doi:10.1016/j.dam.2012.05.023.

3 József Balogh and Robin Pemantle. The Klee-Minty random edge chain moves with linear speed. *Random Structures & Algorithms*, 30(4):464–483, 2007. doi:10.1002/rsa.20127.

4 Yann Disser and Martin Skutella. The simplex algorithm is NP-mighty. 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 858–872. SIAM, 2015. doi:10.1137/1.9781611973730.59.

5 John Fearnley and Rahul Savani. The complexity of the simplex method. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 201–208. ACM, 2015. doi:10.1145/2746539.2746558.

6 John Fearnley and Rahul Savani. The complexity of all-switches strategy improvement. 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 130–139. SIAM, 2016. doi:10.1137/1.9781611974331.ch10.

7 Jan Foniok, Bernd Gärtner, Lorenz Klaus, and Markus Sprecher. Counting unique-sink orientations. *Discrete Applied Mathematics*, 163, Part 2:155–164, 2014. doi:10.1016/j.dam.2013.07.017.

8 Oliver Friedmann. A subexponential lower bound for Zadeh's pivoting rule for solving linear programs and games. In Oktay Günlük and Gerhard J. Woeginger, editors, *Integer*

Programming and Combinatoral Optimization – 15th International Conference, IPCO 2011, New York, NY, USA, June 15-17, 2011. Proceedings, volume 6655 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2011. doi:10.1007/978-3-642-20807-2_16.

9 Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 283–292. ACM, 2011. doi:10.1145/1993636.1993675.

10 Bernd Gärtner. The Random-Facet simplex algorithm on combinatorial cubes. *Random Structures & Algorithms*, 20(3), 2002. doi:10.1002/rsa.10034.

11 Bernd Gärtner, Walter D. Morris Jr., and Leo Rüst. Unique sink orientations of grids. *Algorithmica*, 51(2):200–235, 2008. doi:10.1007/s00453-007-9090-x.

12 Bernd Gärtner and Ingo Schurr. Linear programming and unique sink orientations. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 749–757. ACM Press, 2006. URL: http://dl.acm.org/citation.cfm?id=1109557.1109639.

13 Bernd Gärtner and Antonis Thomas. The complexity of recognizing unique sink orientations. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 341–353. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.341.

14 Bernd Gärtner and Antonis Thomas. The Niceness of Unique Sink Orientations. *CoRR*, June 2016. arXiv:1606.07709.

15 Peter L. Hammer, Bruno Simeone, Thomas M. Liebling, and Dominique de Werra. From linear separability to unimodality: A hierarchy of pseudo-boolean functions. *SIAM J. Discrete Math.*, 1(2):174–184, 1988. doi:10.1137/0401019.

16 Thomas Dueholm Hansen, Mike Paterson, and Uri Zwick. Improved upper bounds for random-edge and random-jump on abstract cubes. 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 874–881. SIAM, 2014. doi:10.1137/1.9781611973402.65.

17 Thomas Dueholm Hansen and Uri Zwick. Random-edge is slower than random-facet on abstract cubes. To appear in ICALP 2016., 2016. URL: http://cs.au.dk/~tdh/papers/Random-Edge-AUSO.pdf.

18 Gil Kalai. A subexponential randomized simplex algorithm (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 475–482. ACM, 1992. doi:10.1145/129712.129759.

19 Lorenz Klaus and Hiroyuki Miyata. Enumeration of PLCP-orientations of the 4-cube. *European Journal of Combinatorics*, 50:138–151, 2015. Combinatorial Geometries: Matroids, Oriented Matroids and Applications. Special Issue in Memory of Michel Las Vergnas. doi:10.1016/j.ejc.2015.03.010.

20 Victor Klee and George J. Minty. How good is the simplex algorithm? *Inequalities III*, pages 159–175, 1972.

21 Jiří Matoušek. Lower bounds for a subexponential optimization algorithm. *Random Structures & Algorithms*, 5(4):591–607, 1994. doi:10.1002/rsa.3240050408.

22 Jiří Matoušek. The number of unique-sink orientations of the hypercube*. *Combinatorica*, 26(1):91–99, 2006. doi:10.1007/s00493-006-0007-0.

23 Jiří Matoušek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4/5):498–516, 1996. doi:10.1007/BF01940877.

24 Jiří Matoušek and Tibor Szabó. RANDOM EDGE can be exponential on abstract cubes. *Advances in Mathematics*, 204(1):262–277, 2006. doi:10.1109/FOCS.2004.56.

25 Walter D. Morris Jr. Randomized pivot algorithms for P-matrix linear complementarity problems. *Mathematical Programming*, 92(2):285–296, 2002. doi:10.1007/s101070100268.

26 Ingo Schurr and Tibor Szabó. Finding the sink takes some time: An almost quadratic lower bound for finding the sink of unique sink oriented cubes. *Discrete & Computational Geometry*, 31(4):627–642, 2004. doi:10.1007/s00454-003-0813-8.

27 Ingo Schurr and Tibor Szabó. Jumping doesn't help in abstract cubes. In Michael Jünger and Volker Kaibel, editors, *Integer Programming and Combinatorial Optimization, 11th International IPCO Conference, 2005*, volume 3509 of *LNCS*, pages 225–235. Springer, 2005. doi:10.1007/11496915_17.

28 Alan Stickney and Layne Watson. Digraph models of Bard-type algorithms for the linear complementarity problem. *Math. Oper. Res.*, 3(4):322–333, 1978. doi:10.1287/moor.3.4.322.

29 Tibor Szabó and Emo Welzl. Unique sink orientations of cubes. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 547–555. IEEE Computer Society, 2001. doi:10.1109/SFCS.2001.959931.

30 Emo Welzl. i-Niceness. http://www.ti.inf.ethz.ch/ew/workshops/01-lc/problems/node7.html, 2001.

31 Kathy Williamson-Hoke. Completely unimodal numberings of a simple polytope. *Discrete Applied Mathematics*, 20(1):69–81, 1988. doi:10.1016/0166-218X(88)90042-X.

Uniqueness, Spatial Mixing, and Approximation for Ferromagnetic 2-Spin Systems

Heng Guo[*1] **and Pinyan Lu**[2]

1 School of Mathematical Sciences, Queen Mary University of London, U.K.
`h.guo@qmul.ac.uk`

2 The Institute for Theoretical Computer Science, School of Information Management and Engineering, Shanghai University of Finance and Economics, China
`lu.pinyan@mail.shufe.edu.cn`

Abstract

For anti-ferromagnetic 2-spin systems, a beautiful connection has been established, namely that the following three notions align perfectly: the uniqueness of Gibbs measures in infinite regular trees, the decay of correlations (also known as spatial mixing), and the approximability of the partition function. The uniqueness condition implies spatial mixing, and an FPTAS for the partition function exists based on spatial mixing. On the other hand, non-uniqueness implies some long range correlation, based on which NP-hardness reductions are built.

These connections for ferromagnetic 2-spin systems are much less clear, despite their similarities to anti-ferromagnetic systems. The celebrated Jerrum-Sinclair Markov chain [8] works even if spatial mixing fails. Also, for a fixed degree the uniqueness condition is non-monotone with respect to the external field, which seems to have no meaningful interpretation in terms of computational complexity. However, it is still intriguing whether there are some relationship underneath the apparent disparities among them.

We provide some answers to this question. Let β, γ be the $(0,0)$ and $(1,1)$ edge interactions respectively ($\beta\gamma > 1$), and λ the external field for spin "0". For graphs with degree bound $\Delta \le \Delta_c + 1$ where $\Delta_c = \frac{\sqrt{\beta\gamma}+1}{\sqrt{\beta\gamma}-1}$, regardless of the field (even inconsistent fields are allowed), correlation decay always holds and FPTAS exists. If all fields satisfy $\lambda < \lambda_c$ (assuming $\beta \le \gamma$), where $\lambda_c = (\gamma/\beta)^{\frac{\Delta_c+1}{2}}$, then a weaker version of spatial mixing holds in all trees. Moreover, if $\beta \le 1$, then $\lambda < \lambda_c$ is sufficient to guarantee strong spatial mixing and FPTAS. This improves the best previous algorithm, a Markov chain based FPRAS for $\lambda \le \gamma/\beta$ [13].

The bound λ_c is almost optimal and can be viewed as a variant of the uniqueness condition with the degree d relaxed to be a real number instead of an integer. When $\beta \le 1$, uniqueness holds in all infinite regular trees, if and only if $\lambda \le \lambda_c^{int}$, where $\lambda_c^{int} = (\gamma/\beta)^{\frac{\lceil\Delta_c\rceil+1}{2}}$. If we allow fields $\lambda > \lambda_c^{int\prime}$, where $\lambda_c^{int\prime} = (\gamma/\beta)^{\frac{\lfloor\Delta_c\rfloor+2}{2}}$, then approximating the partition function is #BIS-hard.

Interestingly, unless Δ_c is an integer, neither λ_c nor λ_c^{int} is the tight bound in each own respect. We provide examples where correlation decay continues to hold in a small interval beyond λ_c, and irregular trees in which spatial mixing fails for some $\lambda < \lambda_c^{int}$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics

Keywords and phrases Approximate counting, Ising model, Spin systems, Correlation decay

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.31

[*] HG is supported by EPSRC Standard Research Grant EP/N004221/1.

1 Introduction

Spin systems model nearest neighbor interactions. In this paper we study 2-state spin systems. An instance is a graph $G = (V, E)$, and a configuration σ assigns one of the two spins "0" and "1" to each vertex; that is, σ is one of the $2^{|V|}$ possible assignments $\sigma : V \to \{0, 1\}$. The local interaction along an edge is specified by a matrix $\mathbf{A} = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$, where $A_{i,j}$ is the (non-negative) local weight when the two endpoints are assigned i and j respectively. We study symmetric edge interactions, that is, $A_{0,1} = A_{1,0}$. Normalize $\mathbf{A}$ so that $\mathbf{A} = \begin{bmatrix} \beta & 1 \\ 1 & \gamma \end{bmatrix}$. Moreover, we also consider the external field, specified by a mapping $\pi : V \to \mathbb{R}^+$. When a vertex is assigned "0", we give it a weight $\pi(v)$. For a particular configuration σ, its weight $w(\sigma)$ is a product over all edge interactions and vertex weights, that is

$$w(\sigma) = \beta^{m_0(\sigma)}\gamma^{m_1(\sigma)} \prod_{v|\sigma(v)=0} \pi(v),$$

where $m_0(\sigma)$ is the number of $(0, 0)$ edges under the configuration σ and $m_1(\sigma)$ is the number of $(1, 1)$ edges. An important special case is the Ising model, where $\beta = \gamma$. The Gibbs measure is a natural distribution in which each configuration σ is drawn with probability proportional to its weight, that is, $\Pr_{G;\beta,\gamma,\pi}(\sigma) \sim w(\sigma)$. The normalizing factor of the Gibbs measure is called the partition function, defined by $Z_{\beta,\gamma,\pi}(G) = \sum_{\sigma:V\to\{0,1\}} w(\sigma)$. The partition function encodes rich information regarding the macroscopic behavior of the spin system. We will be interested in the computational complexity of approximating $Z_{\beta,\gamma,\pi}(G)$. We also simply write $Z_{\beta,\gamma,\lambda}(G)$ when the field is uniform, that is, $\pi(v) = \lambda$ for all $v \in V$. A system with uniform fields is specified by the three parameters (β, γ, λ).

Spin systems not only are interesting in statistical physics, but also find applications in computer science, under the name of *Markov random fields*. In fact, a 2-state spin system is equivalent to a binary Markov random field, and computing the partition function is central to statistical inference. According to their physical and computational properties, spin systems can be classified into two families: *ferromagnetic* systems where the edge interaction is attractive ($\beta\gamma > 1$), and *anti-ferromagnetic* systems where it is repulsive ($\beta\gamma < 1$).

Recently, beautiful connections have been established regarding three different aspects of anti-ferromagnetic 2-spin systems. The uniqueness of Gibbs measures in infinite regular trees[1] of degrees up to Δ implies correlation decay[2] in all graphs of maximum degree Δ, and therefore the existence of fully polynomial-time approximation scheme (FPTAS) for the partition function [19, 11, 17, 12]. On the other hand, if the tree uniqueness fails, then long range correlation appears and the partition function has no fully polynomial-time randomized approximation scheme (FPRAS) unless NP = RP [18, 4]. It suggests that the mathematical property of tree uniqueness, the physical property of spatial mixing, and the computational complexity of approximating the partition function, line up perfectly in anti-ferromagnetic 2-spin systems.

For ferromagnetic systems, the picture is much less clear. In a seminal paper [8], Jerrum and Sinclair gave an FPRAS for the ferromagnetic Ising model $\beta = \gamma > 1$ with any consistent external field λ for general graphs without degree bounds. Thus, there is no computational complexity transition of approximating these models, whereas uniqueness and spatial mixing do exhibit phase transition. This is in sharp contrast to anti-ferromagnetic Ising models

[1] This property is called "tree uniqueness" or "uniqueness" for short. See Sections 2.2 and 6.1 for details.
[2] That is, the correlation of any two vertices decay exponentially in distance. It is also called "spatial mixing".

$\beta = \gamma < 1$, where computational and phase transitions align perfectly. It is not clear at all whether spatial mixing or correlation decay plays any role in the computational complexity.

For more general ferromagnetic 2-spin systems with external fields, the threshold for approximating the partition function is still open. On the complexity side, Goldberg and Jerrum showed that any ferromagnetic 2-spin system is no harder than counting independent sets in bipartite graphs (#BIS) [6], which is conjectured to have no FPRAS [3] (the approximation complexity of #BIS is still open). Based on an earlier result [1], Liu, Lu and Zhang showed that approximating the partition function is #BIS-hard if we allow external fields beyond $(\gamma/\beta)^{\frac{\lfloor\Delta_c\rfloor+2}{2}}$ where $\Delta_c = \frac{\sqrt{\beta\gamma}+1}{\sqrt{\beta\gamma}-1}$ [13].[3]

On the algorithmic side, by reducing to the Ising model, an MCMC based FPRAS is known for the range of $\lambda \le \sqrt{\gamma/\beta}$ [7], which has been recently improved to $\lambda \le \gamma/\beta$ [13]. On the other hand, if we apply the correlation decay algorithmic framework to various pairs of parameters (β, γ), it is not hard to get bounds better than γ/β. However, such success for individual problems does not seem to share meaningful inner connections. In particular, it is not clear how far one can push this method, and to the best of our knowledge, no threshold has even been conjectured.

1.1 Our Contribution

In this paper, we identify a new threshold that almost tightly maps out the boundary of the correlation decay regime, that is, $\lambda_c = (\gamma/\beta)^{\frac{\Delta_c+1}{2}} = (\gamma/\beta)^{\frac{\sqrt{\beta\gamma}}{\sqrt{\beta\gamma}-1}}$. We show that for any $\lambda < \lambda_c$ a variant of spatial mixing holds (Theorem 1) for arbitrary trees. An interesting feature of our work is that we do not restrict the degree or the shape of the tree. This is almost tight since it does not hold if $\lambda > (\gamma/\beta)^{\frac{\lceil\Delta_c\rceil+1}{2}}$. This spatial mixing is weaker than what an algorithm usually requires, but in the regime of $\beta \le 1$ it implies (and therefore is equivalent to) strong spatial mixing. As an algorithmic consequence, we have FPTAS for all $\beta \le 1 < \gamma$, $\beta\gamma > 1$, and $\lambda < \lambda_c$ (Theorem 2). Recall that if we allow λ beyond $(\gamma/\beta)^{\frac{\lfloor\Delta_c\rfloor+2}{2}}$, then the problem is #BIS-hard [13]. Hence only an integral gap remains for the $\beta \le 1 < \gamma$ case.

Formally, let p_v be the marginal probability of v (being assigned "0").

▶ **Theorem 1.** *Let (β, γ, λ) be a set of parameters of the system such that $\beta\gamma > 1$, $\beta \le \gamma$, and $\lambda < \lambda_c$. Let T_v and $T'_{v'}$ be two trees with roots v and v' respectively. If the two trees have the same structure in the first ℓ levels, then $|p_v - p_{v'}| \le O(\exp(-\ell))$.*

In other words, if we simply truncate a tree at depth ℓ, the marginal probability of its root will change by only at most $O(\exp(-\ell))$. Surprisingly, if we replace λ_c by its integral counterpart, then this implication no longer holds and there is a counterexample (see Section 5). More precisely, it is no longer true that the uniqueness in infinite regular trees implies correlation decay in graphs or even trees, since our counterexample is an irregular tree. We note that this is in sharp contrast to anti-ferromagnetic systems, where (integral) uniqueness implies correlation decay.

From the computational complexity point of view, we would like to get FPTAS for the partition function, which requires a condition called strong spatial mixing (SSM). It is stronger than the spatial mixing established in Theorem 1 by imposing arbitrary partial configurations. We are able to prove SSM with $\lambda < \lambda_c$ for the range of $\beta \le 1$. Indeed, if

[3] Here and below we assume $\beta \le \gamma$ due to symmetry.

$\beta \leq 1$, then the two versions of spatial mixing are equivalent. Let I be an interval of the form $[\lambda_1, \lambda_2]$ or $(\lambda_1, \lambda_2]$. We consider the following problem.

Name: $\#2\textsc{Spin}(\beta, \gamma, I)$
Instance: A graph $G = (V, E)$ and a mapping $\pi : V \to \mathbb{R}^+$, such that $\pi(v) \in I$ for any $v \in V$.
Output: $Z_{\beta,\gamma,\pi}(G)$.

Then we have the following theorem.

▶ **Theorem 2.** *Let (β, γ, λ) be a set of parameters of the system such that $\beta\gamma > 1$, $\beta \leq 1$ and $\lambda < \lambda_c$. Then $\#2\textsc{Spin}(\beta, \gamma, (0, \lambda])$ has an FPTAS.*

Therefore, we get an almost tight dichotomy for ferromagnetic 2-spin systems when $\beta \leq 1$, since $\#2\textsc{Spin}(\beta, \gamma, (0, \lambda])$ is #BIS-hard, if λ is larger than the integral counterpart of λ_c [13] (see also Proposition 22).

The reason behind λ_c is a nice interplay among uniqueness, spatial mixing, and approximability. We start with some purely mathematical observations on the symmetric tree recursion $f_d(x) = \lambda \left(\frac{\beta x+1}{x+\gamma}\right)^d$, an increasing function in x. Relax the range of d in $f_d(x)$ to be real numbers. Then Δ_c is the critical (possibly fractional) degree and λ_c is the corresponding critical external field for the recursion to have a unique fixed point. This set of critical parameters enjoys some very nice mathematical properties. For $d = \Delta_c$ and $\lambda = \lambda_c$, the function $f_d(x)$ has a unique fixed point $\widehat{x} = \sqrt{\gamma/\beta}$ and $f'_d(\widehat{x}) = 1$. Moreover, it also satisfies that $f''_d(\widehat{x}) = 0$, which is a necessary condition for the contraction of the tree recursion (easily derived using the heuristic of finding potential functions described in [12]). All these nice mathematical properties prove to be useful in our later analysis. For degrees other than Δ_c, their critical external fields are much less convenient — the function $f_d(x)$ has two fixed points: one is crossing and the other is tangent. Moreover, $f''_d(\hat{x}) = 0$ does not necessarily hold.

The proof of Theorem 1 uses the potential method to analyze decay of correlation, which is now streamlined (see e.g. [12]). The main difficulty is to find a good potential function. In other words, we want to solve a variational problem minimizing the maximum of the decay rate function. The main novelty in our solution is that we restrict variables to the range of $(0, \frac{\lambda}{1+\lambda}]$ and our potential function is well-defined only in this range. This is in fact necessary, as otherwise the statement does not hold, and is valid for the setting of Theorem 1. Also note that with our choice, the proof is relatively clean and significantly simpler than similar proofs in other settings. In particular, we do not need the "symmetrization" argument (see e.g. [12, 16]). We also use a trick of truncating the potential to deal with unbounded degrees (see Eq. (5)).

For the range of $\beta > 1$, SSM does not hold even if $\lambda < \lambda_c$. However, we conjecture that Theorem 2 can be extended to the $\beta > 1$ range as well, mainly due to Theorem 1, which does not require $\beta \leq 1$. Moreover, we show that even if $\beta > 1$, the marginal probability in any instance is within the range of $(0, \frac{\lambda}{1+\lambda}]$ given $\lambda < \lambda_c$ (see Proposition 20). This seems to imply that the main reason why our algorithm fails is due to pinnings (forcing a vertex to be "0" or "1") in the self-avoiding walk tree construction, whereas in a real instance these pinnings cannot aggregate enough "bad" influence. However, to turn such intuition into an algorithm requires a careful treatment of these pinnings to achieve an FPTAS without SSM. We leave this as an important open question.

At last, we note that neither λ_c nor its integral counterpart is the exact threshold in each own respect, even if $\beta \leq 1$. Strong spatial mixing continues to hold even if $\lambda > \lambda_c$ in a small interval. We give a concrete example to illustrate this point in Section 4, Proposition 21. Moreover, as mentioned earlier, an irregular tree exists where the correlation decay threshold is lower than the threshold for all infinite regular trees. This is discussed in Section 5. It is another important open question to figure out the exact threshold between λ_c and its integral counterpart(s).

2 Preliminaries

An instance of a 2-spin system is a graph $G = (V, E)$. A configuration $\sigma : V \to \{0, 1\}$ assigns one of the two spins "0" and "1" to each vertex. We normalize the edge interaction to be $\begin{bmatrix} \beta & 1 \\ 1 & \gamma \end{bmatrix}$, and also consider the external field, specified by a mapping $\pi : V \to \mathbb{R}^+$. When a vertex is assigned "0", we give it a weight $\pi(v)$. All parameters are non-negative. For a particular configuration σ, its weight $w(\sigma)$ is a product over all edge interactions and vertex weights, that is

$$w(\sigma) = \beta^{m_0(\sigma)}\gamma^{m_1(\sigma)} \prod_{v|\sigma(v)=1} \pi(v), \tag{1}$$

where $m_0(\sigma)$ is the number of $(0, 0)$ edges given by the configuration σ and $m_1(\sigma)$ is the number of $(1, 1)$ edges. An important special case is the Ising model, where $\beta = \gamma$. Notice that in the statistic physics literature, parameters are usually chosen to be the logarithms of our parameters above. Different parameterizations do not affect the complexity of the same system.

We also write $\lambda_v := \pi(v)$. If π is a constant function such that $\lambda_v = \lambda > 0$ for all $v \in V$, we also denote it by λ. We say π has a lower bound (or an upper bound) $\lambda > 0$, if π satisfies the guarantee that $\lambda_v \geq \lambda$ (or $\lambda_v \leq \lambda$).

The Gibbs measure is a natural distribution in which each configuration σ is drawn with probability proportional to its weight, that is, $\Pr_{G;\beta,\gamma,\pi}(\sigma) \sim w(\sigma)$. The normalizing factor of the Gibbs measure is called the partition function, defined by $Z_{\beta,\gamma,\pi}(G) = \sum_{\sigma:V\to\{0,1\}} w(\sigma)$. Recall that we are interested in the computational problem #2Spin(β, γ, I), where I is an interval of the form $[\lambda_1, \lambda_2]$ or $(\lambda_1, \lambda_2]$, for which $Z_{\beta,\gamma,\pi}(G)$ is the output. When input graphs are restricted to have a degree bound Δ, we write #Δ-2Spin(β, γ, I) to denote the problem. When the field is uniform, that is, λ is the only element in I, we simply write #2Spin(β, γ, λ). Due to [2] and a standard diagonal transformation, for any constant $\lambda > 0$, #2Spin(β, γ, λ) is #P-hard unless $\beta = \gamma = 0$ or $\beta\gamma = 1$.

2.1 The Self-Avoiding Walk Tree

We briefly describe Weitz's algorithm [19]. Our algorithms presented later will follow roughly the same paradigm.

The Gibbs measure defines a marginal distribution of spins for each vertex. Let p_v denote the probability of a vertex v being assigned "0". Since the system is self-reducible, #2Spin(β, γ, λ) is equivalent to computing p_v for any vertex v [9] (for details, see for example Lemma 8).

Let $\sigma_\Lambda \in \{0, 1\}^\Lambda$ be a configuration of $\Lambda \subset V$. We call vertices in Λ *fixed* and other vertices *free*. We use $p_v^{\sigma_\Lambda}$ to denote the marginal probability of v being assigned "0" conditional on the configuration σ_Λ of Λ.

Suppose the instance is a tree T with root v. Let $R_T^{\sigma_\Lambda} := p_v^{\sigma_\Lambda}/(1 - p_v^{\sigma_\Lambda})$ be the ratio between the two probabilities that the root v is 0 and 1, while imposing some condition σ_Λ

(with the convention that $R_T^{\sigma_\Lambda} = \infty$ when $p_v^{\sigma_\Lambda} = 1$). Suppose that v has d children $v_i, \dots v_d$. Let T_i be the subtree with root v_i. Due to the independence of subtrees, it is straightforward to get the following recursion for calculating $R_T^{\sigma_\Lambda}$:

$$R_T^{\sigma_\Lambda} = F_d\left(R_{T_1}^{\sigma_\Lambda}, \dots, R_{T_d}^{\sigma_\Lambda}\right), \tag{2}$$

where the function $F_d(x_1, \dots, x_d)$ is defined as

$$F_d(x_1, \dots, x_d) := \lambda_v \prod_{i=1}^{d} \frac{\beta x_i + 1}{x_i + \gamma}.$$

We allow x_i's to take the value ∞ as in that case the function F_d is clearly well defined. In general we use capital letters like $F, G, C, \dots$ to denote multivariate functions, and small letters $f, g, c, \dots$ to denote their symmetric versions, where all variables take the same value. Here we define $f_d(x) := \lambda\left(\frac{\beta x+1}{x+\gamma}\right)^d$ to be the symmetric version of $F_d(\mathbf{x})$.

Let $G(V, E)$ be a graph. Similarly define $R_{G,v}^{\sigma_\Lambda} := p_v^{\sigma_\Lambda}/(1 - p_v^{\sigma_\Lambda})$. In contrast to the case of trees, there is no easy recursion to calculate $R_{G,v}^{\sigma_\Lambda}$ for a general graph G. This is because of dependencies introduced by cycles. Weitz [19] reduced computing the marginal distribution of v in a general graph G to that in a tree, called the self-avoiding walk (SAW) tree, denoted by $T_{\text{SAW}}(G, v)$. To be specific, given a graph $G = (V, E)$ and a vertex $v \in V$, $T_{\text{SAW}}(G, v)$ is a tree with root v that enumerates all self-avoiding walks originating from v in G, with additional vertices closing cycles as leaves of the tree. Each vertex in the new vertex set V_{SAW} of $T_{\text{SAW}}(G, v)$ corresponds to a vertex in G, but a vertex in G may be mapped to more than one vertices in V_{SAW}. A boundary condition is imposed on leaves in V_{SAW} that close cycles. The imposed colors of such leaves depend on whether the cycle is formed from a small vertex to a large vertex or conversely, where the ordering is arbitrarily chosen in G. Vertex sets $S \subset \Lambda \subset V$ are mapped to respectively $S_{\text{SAW}} \subset \Lambda_{\text{SAW}} \subset V_{\text{SAW}}$, and any configuration $\sigma_\Lambda \in \{0,1\}^\Lambda$ is mapped to $\sigma_{\Lambda_{\text{SAW}}} \in \{0,1\}^{\Lambda_{\text{SAW}}}$. With slight abuse of notations we may write $S = S_{\text{SAW}}$ and $\sigma_\Lambda = \sigma_{\Lambda_{\text{SAW}}}$ when no ambiguity is caused.

▶ **Proposition 3** (Theorem 3.1 of Weitz [19]). *Let $G = (V, E)$ be a graph, $v \in V$, $\sigma_\Lambda \in \{0,1\}^\Lambda$ be a configuration on $\Lambda \subset V$, and $S \subset V$. Let $T = T_{\text{SAW}}(G, v)$ be constructed as above. It holds that*

$$R_{G,v}^{\sigma_\Lambda} = R_T^{\sigma_\Lambda}.$$

Moreover, the maximum degree of T is at most the maximum degree of G, $\text{dist}_G(v, S) = \text{dist}_T(v, S_{SAW})$, and any neighborhood of v in T can be constructed in time proportional to the size of the neighborhood.

The SAW tree construction does not solve a #P-hard problem, since $T_{\text{SAW}}(G, v)$ is potentially exponentially large in size of G. For a polynomial time approximation algorithm, we may run the tree recursion within some polynomial size, or equivalently a logarithmic depth. At the boundary where we stop, we plug in some arbitrary values. The question is then how large is the error due to our random guess. To guarantee the performance of the algorithm, we need the following notion of *strong spatial mixing.*

▶ **Definition 4.** A spin system on a family $\mathcal{G}$ of graphs is said to exhibit *strong spatial mixing* (SSM) if for any graph $G = (V, E) \in \mathcal{G}$, any $v \in V, \Lambda \subset V$ and any $\sigma_\Lambda, \tau_\Lambda \in \{0,1\}^\Lambda$,

$$|p_v^{\sigma_\Lambda} - p_v^{\tau_\Lambda}| \le \exp(-\Omega(\text{dist}(v, S))),$$

where $S \subset \Lambda$ is the subset on which σ_Λ and τ_Λ differ, and $\text{dist}(v, S)$ is the shortest distance from v to any vertex in S.

Weak spatial mixing is defined similarly by measuring the decay with respect to $\mathrm{dist}(v, \Lambda)$ instead of $\mathrm{dist}(v, S)$. Spatial mixing properties are also called correlation decay in statistical physics.

If SSM holds, then the error caused by early termination in $T_{\mathrm{SAW}}(G, v)$ and arbitrary boundary values is only exponentially small in the depth. Hence the algorithm is an FPTAS. In a lot of cases, the existence of an FPTAS boils down to establish SSM.

2.2 The Uniqueness Condition in Regular Trees

Let $\mathbb{T}_d$ denote the infinite d-regular tree, also known as the *Bethe lattice* or the *Cayley tree*. If we pick an arbitrary vertex as the root of $\mathbb{T}_d$, then the root has d children and every other vertex has $d-1$ children. Notice that the difference between $\mathbb{T}_d$ and an infinite $(d-1)$-ary tree is only the degree of the root. We consider the uniqueness of Gibbs measures on $\mathbb{T}_d$, where the field is uniformly $\lambda > 0$. Due to the symmetric structure of $\mathbb{T}_d$, the standard recursion (2) thus becomes $R_v = f_{d-1}(R_{v_i})$ (for any vertex v other than the root), where $f_d(x) = \lambda \left(\frac{\beta x+1}{x+\gamma}\right)^d$ is the symmetrized version of $F_d(\mathbf{x})$.

For anti-ferromagnetic systems, that is, $\beta\gamma < 1$, there is a unique fixed point to $f_d(x) = x$, denoted by $\widehat{x}$. It has been shown that the Gibbs measure in $\mathbb{T}_d$ is unique if and only if $\left|f'_{d-1}(\widehat{x})\right| \le 1$ [10, 5].

In contrast, if $\beta\gamma > 1$, then $f'_d(x) > 0$ for any $x > 0$. There may be 1 or 3 positive fixed points such that $x = f_d(x)$. It is known [10, 5] that the Gibbs measure of two-state spin systems in $\mathbb{T}_d$ is unique if and only if there is only one fixed point for $x = f_{d-1}(x)$, or equivalently, for all fixed points $\widehat{x}_d$ of $f_d(x)$, $f'_d(\widehat{x}_d) < 1$.

Let $\Delta_c := \frac{\sqrt{\beta\gamma}+1}{\sqrt{\beta\gamma}-1}$. Then we have the following result.

▶ **Proposition 5.** *If $\Delta - 1 < \Delta_c$, then the uniqueness condition in $\mathbb{T}_\Delta$ holds regardless of the field.*

Note that the condition $\Delta - 1 < \Delta_c$ matches the exact threshold of fast mixing for Gibbs samplers in the Ising model [15]. In Section 3.1, we will show that, SSM holds and there exists an FPTAS for the partition function, in graphs with degree bound $\Delta < \Delta_c + 1$. This is Theorem 13.

To study general graphs, one needs to consider infinite regular trees of all degrees. If $\beta > 1$ (still assuming $\beta\gamma > 1$ and $\beta \le \gamma$), then there is no λ such that the uniqueness condition holds in $\mathbb{T}_d$ for all degrees $d \ge 2$. In contrast, let $\lambda_c^{int} := (\gamma/\beta)^{\frac{\lceil\Delta_c\rceil+1}{2}}$ and we have the following.

▶ **Proposition 6.** *Let (β, γ) be two parameters such that $\beta\gamma > 1$ and $\beta \le 1 < \gamma$. The uniqueness condition holds in $\mathbb{T}_d$ for all degrees $d \ge 2$ if and only if $\lambda < \lambda_c^{int}$.*

However, there exists (β, γ, λ) and an (irregular) tree T such that $\beta\gamma > 1$, $\beta \le 1 < \gamma$, and $\lambda < \lambda_c^{int}$ and SSM does not hold in T. This is discussed in Section 5. Recall that $\lambda_c := (\gamma/\beta)^{\frac{\Delta_c+1}{2}}$. If we replace λ_c^{int} with $\lambda_c \le \lambda_c^{int}$ in the condition of Proposition 6, that is, $\beta\gamma > 1$, $\beta \le 1 < \gamma$, and $\lambda < \lambda_c$, then SSM holds in all graphs and an FPTAS exists. This is shown in Section 3.2, Theorem 18.

Details and proofs about Propositions 5 and 6 are given in Section 6.1.

2.3 The Potential Method

We would like to prove the strong spatial mixing in arbitrary trees, sometimes with bounded degree Δ, under certain conditions. This is sufficient for approximation algorithms due to the self-avoiding walk tree construction. Our main technique in the analysis is the potential method. The analysis in this section is a standard routine, with some specialization to ferromagnetic 2-spin models (cf. [12, 16]). To avoid interrupting the flow, we move all details and proofs to Section 6.2.

Roughly speaking, instead of studying (2) directly, we use a potential function $\Phi(x)$ to map the original recursion to a new domain (see the commutative diagram Figure 1). Morally we can choose whatever function as the potential function. However, we would like to pick "good" ones so as to help the analysis of the contraction. Define $\varphi(x) := \Phi'(x)$ and

$$C_{\varphi,d}(\mathbf{x}) := \varphi(F_d(\mathbf{x})) \cdot \sum_{i=1}^{d} \left| \frac{\partial F_d}{\partial x_i} \right| \frac{1}{\varphi(x_i)}.$$

▶ **Definition 7.** Let $\Phi : \mathbb{R}^+ \to \mathbb{R}^+$ be a differentiable and monotonically increasing function. Let $\varphi(x)$ and $C_{\varphi,d}(\mathbf{x})$ be defined as above. Then $\Phi(x)$ is a *good potential function for degree d and field λ* if it satisfies the following conditions:

1. there exists a constant $C_1, C_2 > 0$ such that $C_1 \le \varphi(x) \le C_2$ for all $x \in [\lambda\gamma^{-d}, \lambda\beta^d]$;
2. there exists a constant $\alpha < 1$ such that $C_{\varphi,d}(\mathbf{x}) \le \alpha$ for all $x_i \in [\lambda\gamma^{-d}, \lambda\beta^d]$.

We say $\Phi(x)$ is a good potential function for d and field π, if $\Phi(x)$ is a good potential function for d and any λ in the codomain of π,

In Definition 7, Condition 1 is rather easy to satisfy. The crux is in fact Condition 2. We call α in Condition 2 the amortized contraction ratio of $\Phi(x)$. It has the following algorithmic implication. The proof is based on establishing strong spatial mixing.

▶ **Lemma 8.** *Let (β, γ) be two parameters such that $\beta\gamma > 1$. Let $G = (V, E)$ be a graph with a maximum degree Δ and n many vertices and π be a field on G. Let $\lambda = \max_{v \in V}\{\pi(v)\}$. If there exists a good potential function for π and all $d \in [1, \Delta - 1]$ with contraction ratio $\alpha < 1$, then $Z_{\beta,\gamma,\pi}(G)$ can be approximated deterministically within a relative error ε in time $O\left(n \left(\frac{n\lambda}{\varepsilon}\right)^{\frac{\log(\Delta-1)}{-\log\alpha}}\right)$.*

When the degree is unbounded, the SAW tree may grow super polynomially even if the depth is of order $\log n$. We use a refined metric replacing the naive graph distance used in Definition 4. Strong spatial mixing under this metric is also called *computationally efficient correlation decay* [11, 12].

▶ **Definition 9.** Let T be a rooted tree and $M > 1$ be a constant. For any vertex v in T, define the *M-based depth* of v, denoted $\ell_M(v)$, such that $\ell_M(v) = 0$ if v is the root, and $\ell_M(v) = \ell_M(u) + \lceil \log_M(d+1) \rceil$ if v is a child of u and u has degree d.

Let $B(\ell)$ be the set of all vertices whose M-based depths of v is at most ℓ. It is easy to verify inductively such that $|B(\ell)| \le M^\ell$ in a tree. We then define a slightly stronger notion of potential functions.

▶ **Definition 10.** Let $\Phi : \mathbb{R}^+ \to \mathbb{R}^+$ be a differentiable and monotonically increasing function. Let $\varphi(x)$ and $C_{\varphi,d}(\mathbf{x})$ defined in the same way as in Definition 7. Then $\Phi(x)$ is a *universal potential function* for the field λ if it satisfies the following conditions:

1. there are two constants $C_1, C_2 > 0$ such that $C_1 \le \varphi(x) \le C_2$ for any $x \in (0, \lambda]$;
2. there exists a constant $\alpha < 1$ such that for all d, $C_{\varphi,d}(\mathbf{x}) \le \alpha^{\lceil \log_M(d+1) \rceil}$ for all $x_i \in (0, \lambda]$;

We say $\Phi(x)$ is a universal potential function for a field π, if $\Phi(x)$ is a universal potential function for any λ in the codomain of π. We also call α the contraction ratio and call M the base. The following two lemmas show that our main theorems follow from the existence of a universal potential function.

The way we define universal potential functions restricts them to only apply to the range of $(0, \lambda]$. This will be true in our applications (see for example Claim 16).

▶ **Lemma 11.** *Let (β, γ, λ) be three parameters such that $\beta\gamma > 1$, $\beta \le \gamma$, and $\lambda < \lambda_c$. Let T and T' be two trees that agree on the first ℓ levels with root v and v' respectively. If there exists a universal potential function $\Phi(x)$, then $|p_v - p_{v'}| \le O(\exp(-\ell))$.*

▶ **Lemma 12.** *Let (β, γ) be two parameters such that $\beta\gamma > 1$ and $\beta \le 1 < \gamma$. Let $G = (V, E)$ be a graph with n many vertices and π be a field on G. Let $\lambda = \max_{v \in V}\{\pi(v)\}$. If there exists a universal potential function $\Phi(x)$ for π with contraction ratio $\alpha < 1$ and base M, then $Z_{\beta,\gamma,\pi}(G)$ can be approximated deterministically within a relative error ε in time $O\left(n^3 \left(\frac{n\lambda}{\varepsilon}\right)^{\frac{\log M}{-\log \alpha}}\right)$.*

3 Correlation Decay below Δ_c or λ_c

In this section, we show our main results. We will first show a folklore result for bounded degree graphs with a very simple proof. Then we continue to show the main theorem regarding general graphs. We carefully choose two appropriate potential functions and then apply Lemma 8 or Lemma 12.

3.1 Bounded Degree Graphs

We first apply our framework to get FPTAS for graphs with degree bound $\Delta < \Delta_c + 1 = \frac{2\sqrt{\beta\gamma}}{\sqrt{\beta\gamma}-1}$. Correlation decay for graphs with such degree bounds is folklore and can be found in [14] for the Ising model. Algorithmic implications are also shown, e.g. in [20]. As we shall see, the proof is very simple in our framework. Note that λ, Δ, and α are considered constants for the FPTAS.

▶ **Theorem 13.** *Let (β, γ) be two parameters such that $\beta\gamma > 1$. Let $G = (V, E)$ be a graph with a maximum degree $\Delta < \Delta_c + 1$ and n many vertices, and let π be a field on G. Let $\lambda = \max_{v \in V}\{\pi(v)\}$. Then $Z_{\beta,\gamma,\pi}(G)$ can be approximated deterministically within a relative error ε in time $O\left(n \left(\frac{n\lambda}{\varepsilon}\right)^{\frac{\log(\Delta-1)}{-\log \alpha}}\right)$, where $\alpha = \frac{\Delta-1}{\Delta_c}$.*

Proof. We choose our potential function to be $\Phi_1(x) = \log x$ such that $\varphi_1(x) := \Phi_1'(x) = \frac{1}{x}$. We verify the conditions of Definition 7. Condition 1 is trivial. For Condition 2, we have that for any integer $1 \le d \le \Delta - 1$,

$$\begin{aligned} C_{\varphi_1,d}(\mathbf{x}) &= \varphi_1(F_d(\mathbf{x})) \sum_{i=1}^{d} \frac{\partial F_d}{\partial x_i} \cdot \frac{1}{\varphi_1(x)} \\ &= \frac{1}{F_d(\mathbf{x})} \sum_{i=1}^{d} F_d(\mathbf{x}) \cdot \frac{\beta\gamma - 1}{(x_i + \beta)(\gamma x_i + 1)} \cdot x_i \\ &= \sum_{i=1}^{d} \frac{(\beta\gamma - 1)x_i}{(\gamma x_i + 1)(x_i + \beta)} \le \sum_{i=1}^{d} \frac{1}{\Delta_c} = \frac{d}{\Delta_c} \le \frac{\Delta - 1}{\Delta_c} = \alpha, \end{aligned}$$

where we used the fact that for any $x > 0$,

$$\frac{(\beta\gamma - 1)x}{(\gamma x + 1)(x + \beta)} \le \frac{1}{\Delta_c}.$$

Hence $\Phi_1(x)$ is a good potential function for all degrees $d \in [1, \Delta - 1]$ with contraction ratio α. The theorem follows by Lemma 8. ◀

Note that Theorem 13 matches the uniqueness condition in Proposition 5 and, restricted to the Ising model, the fast mixing bound of Gibbs samplers in [15].

3.2 General Graphs

Recall that $\lambda_c = \left(\frac{\gamma}{\beta}\right)^{\frac{\Delta_c+1}{2}} = \left(\frac{\gamma}{\beta}\right)^{\frac{\sqrt{\beta\gamma}}{\sqrt{\beta\gamma}-1}}$. The following two technical lemmas show some important properties regarding the threshold λ_c, which are keys to get our main theorems. Proofs are given in Section 6.3.

▶ **Lemma 14.** *Let β, γ be two parameters such that $\beta\gamma > 1$ and $\beta \le \gamma$. For any $0 < x \le \lambda_c$, $\frac{\beta x+1}{x+\gamma} \le 1$.*

▶ **Lemma 15.** *Let β, γ be two parameters such that $\beta\gamma > 1$ and $\beta \le \gamma$. For any $0 < x \le \lambda_c$, we have*

$$(\beta\gamma - 1)x \log \frac{\lambda_c}{x} \le (\beta x + 1)(x + \gamma) \log \frac{x + \gamma}{\beta x + 1}. \tag{3}$$

In our applications, the quantity x in both lemmas will be the ratio of marginal probabilities in trees, denoted by R_v for a vertex v. To make use of these properties, one key requirement is that $0 < x \le \lambda_c$. This is not necessarily true in trees with pinning (and therefore not true in general SAW trees). Nevertheless, it does hold in trees without pinning.

▶ **Claim 16.** *For (β, γ, λ) where $\beta\gamma > 1$, $\beta \le \gamma$, and $\lambda < \lambda_c$, $R_v \in (0, \lambda]$ holds in trees without pinning.*

We prove Claim 16 by induction. For any tree T_v, if v is the only vertex, then $R_v = \lambda$ and the base case holds. Given Lemma 14 and $\lambda < \lambda_c$, the inductive step to show Claim 16 follows from the standard tree recursion (2).

In addition, it also holds when $\beta \le 1$, in trees even with pinning (but not counting the pinned vertices). This includes the SAW tree construction as special cases. To see that, for any vertex v, if one of v's child, say u, is pinned to 0 (or 1), then we can just remove u and change the field of v from λ_v to $\lambda'_v = \lambda_v\beta$ (or $\lambda'_v = \lambda_v/\gamma$), without affecting the marginal probability of v and any other vertices. By our assumptions $\lambda_v < \lambda_c$ and $\beta \le 1 < \gamma$, we have that $\lambda'_v < \lambda_c$ as well. Hence, after removing all pinned vertices, we still have that $\lambda_v \le \lambda_c$ for all $v \in V$. This reduces to Claim 16.

Indeed, both of Theorem 1 and 2 can be generalized to the setting where vertices may have different external fields as long as they are all below λ_c, as follows.

▶ **Theorem 17.** *Let (β, γ) be two parameters such that $\beta\gamma > 1$, $\beta \le \gamma$, and $\lambda < \lambda_c$. Let T_v and $T'_{v'}$ be two trees with roots v and v' respectively. Let $\lambda = \max_{u \in T_v \cup T'_{v'}} \{\pi(u)\}$. If $\lambda < \lambda_c$ and in the first ℓ levels, T_v and $T'_{v'}$ have the same structure and external fields for corresponding pairs of vertices, then $|p_v - p_{v'}| \le O(\exp(-\ell))$.*

▶ **Theorem 18.** *Let (β,γ) be two parameters such that $\beta\gamma > 1$ and $\beta \le 1 < \gamma$. Let $G = (V, E)$ be a graph with n many vertices, and let π be a field on G. Let $\lambda = \max_{v\in V}\{\pi(v)\}$. If $\lambda < \lambda_c$, then $Z_{\beta,\gamma,\pi}(G)$ can be approximated deterministically within a relative error ε in time $O\left(n\left(\frac{n\lambda}{\varepsilon}\right)^{\frac{\log M}{-\log\alpha}}\right)$, where $M > 1$ and $\alpha < 1$ are two constants depending on (β,γ,λ).*

To show Theorem 17 and Theorem 18, we will apply Lemma 11 and Lemma 12. Essentially we only need to show the existence of a universal potential function.

Let $g_\lambda(x) := \frac{(\beta\gamma-1)x\log\frac{\lambda}{x}}{(\beta x+1)(x+\gamma)\log\frac{x+\gamma}{\beta x+1}}$. By Lemma 15, $g_{\lambda_c}(x) \le 1$. For $\lambda < \lambda_c$, note that $\lim_{x\to 0} g_\lambda(x) = 0$. Hence there exists $0 < \varepsilon < \lambda$ and $0 < \delta < 1$ such that if $0 < x < \varepsilon$, $g_\lambda(x) < \delta$. Moreover, if $\varepsilon \le x \le \lambda$, then $\frac{g_\lambda(x)}{g_{\lambda_c}(x)} = \frac{\log\lambda-\log x}{\log\lambda_c-\log x} \le \frac{\log\lambda-\log\varepsilon}{\log\lambda_c-\log\varepsilon}$. Let

$$\alpha_\lambda := \max\left\{\delta, \frac{\log\lambda-\log\varepsilon}{\log\lambda_c-\log\varepsilon}\right\} < 1.$$

Then we have the following lemma.

▶ **Lemma 19.** *Let β,γ be two parameters such that $\beta\gamma > 1$ and $\beta \le \gamma$. If $\lambda < \lambda_c$, then $g_\lambda(x) \le \alpha_\lambda$ for any $0 < x \le \lambda$, where $\alpha_\lambda < 1$ is defined above.*

Let $t := \frac{\alpha_\lambda\gamma}{\beta\gamma-1}\log\frac{\lambda+\gamma}{\beta\lambda+1}$ so that for any $0 < x \le \lambda$,

$$t \le \frac{\alpha_\lambda(\beta x+1)(x+\gamma)}{\beta\gamma-1}\log\frac{x+\gamma}{\beta x+1}.$$

We define $\varphi_2(x) := \min\left\{\frac{1}{t}, \frac{1}{x\log\frac{\lambda}{x}}\right\}$. To be specific, note that $x\log\frac{\lambda}{x} \le \frac{\lambda}{e}$ for any $0 < x \le \lambda$. If $t \ge \frac{\lambda}{e}$, then $\frac{1}{x\log\frac{\lambda}{x}} \ge \frac{1}{t}$ for any $0 < x \le \lambda$. In this case, we let

$$\varphi_2(x) := \frac{1}{t}. \tag{4}$$

Otherwise $t < \frac{\lambda}{e}$, and there are two roots to $x\log\frac{\lambda}{x} = t$ in $(0,\lambda]$. Denote them by x_0 and x_1. We define

$$\varphi_2(x) := \begin{cases} \frac{1}{t} & 0 \le x < x_0; \\ \frac{1}{x\log\frac{\lambda}{x}} & x_0 \le x < x_1; \\ \frac{1}{t} & x_1 \le x < \lambda. \end{cases} \tag{5}$$

We define $\Phi_2(x) := \int_0^x \varphi_2(y)dy$ so that $\Phi_2'(x) = \varphi_2(x)$. By our choice of $\varphi_2(x)$, it always holds that for any $0 < x \le \lambda$,

$$\varphi_2(x)x\log\frac{\lambda}{x} \le 1, \tag{6}$$

and by Lemma 14 and Lemma 19,

$$\frac{\beta\gamma-1}{(\beta x+1)(x+\gamma)}\cdot\frac{1}{\varphi_2(x)} \le \alpha_\lambda\log\frac{x+\gamma}{\beta x+1}. \tag{7}$$

Now, we are ready to prove Theorems 17 and 18.

Proof of Theorems 17 and 18. We claim that $\Phi_2(x)$ is a universal potential function for any field π with an upper bound λ, with contraction ratio α_λ given above and base M that will be determined shortly. Theorem 17 and Theorem 18 follow from $\Phi_2(x)$ combined with Lemma 11 and 12, respectively. We verify the two conditions in Definition 10.

For Condition 1, it is easy to see that in case (4), $\varphi_2(x) = \frac{1}{t}$ for any $x \in (0, \lambda]$, and in case (5), $\frac{e}{\lambda} \le \varphi_2(x) \le \frac{1}{t}$ for any $x \in (0, \lambda]$.

For Condition 2, we have that

$$\begin{aligned}
C_{\varphi_2,d}(\mathbf{x}) &= \varphi_2(F_d(\mathbf{x})) \sum_{i=1}^{d} \frac{\partial F_d}{\partial x_i} \cdot \frac{1}{\varphi_2(x_i)} \\
&= \varphi_2(F_d(\mathbf{x}))F_d(\mathbf{x}) \sum_{i=1}^{d} \frac{\beta\gamma - 1}{(\beta x_i + 1)(x_i + \gamma)} \cdot \frac{1}{\varphi_2(x_i)} \\
&\le \varphi_2(F_d(\mathbf{x}))F_d(\mathbf{x}) \sum_{i=1}^{d} \alpha_\lambda \log \frac{x_i + \gamma}{\beta x_i + 1} && \text{(by (7))} \\
&= \alpha_\lambda \varphi_2(F_d(\mathbf{x}))F_d(\mathbf{x}) \log \frac{\lambda}{F_d(\mathbf{x})} \\
&\le \alpha_\lambda. && \text{(by (6))}
\end{aligned}$$

Moreover, $F_d(\mathbf{x}) < \lambda \left(\frac{\beta\lambda+1}{\lambda+\gamma}\right)^d$ for any $x_i \in (0, \lambda]$, and $\frac{\beta\lambda+1}{\lambda+\gamma} < 1$ by Lemma 14. Then there exists $d_0 \ge 1$ such that $\left(\frac{\beta\lambda+1}{\lambda+\gamma}\right)^{d_0} < e^{-1}$. Hence, for any $d > d_0$,

$$\begin{aligned}
C_{\varphi_2,d}(\mathbf{x}) &\le \frac{\alpha_\lambda}{t} F_d(\mathbf{x}) \log \frac{\lambda}{F_d(\mathbf{x})} \\
&\le \frac{\alpha_\lambda \lambda}{t} \left(\frac{\beta\lambda + 1}{\lambda + \gamma}\right)^d d \log \frac{\beta\lambda + 1}{\lambda + \gamma}.
\end{aligned}$$

Therefore, there exists an integer $M \ge d_0$ such that for any $1 \le d < M$, $C_{\varphi_2,d}(\mathbf{x}) \le \alpha_\lambda \le \alpha_\lambda^{\lceil \log_M(d+1) \rceil}$ and for any $d \ge M$, $C_{\varphi_2,d}(\mathbf{x}) \le \frac{\alpha_\lambda \lambda}{t} \left(\frac{\beta\lambda+1}{\lambda+\gamma}\right)^d d \log \left(\frac{\beta\lambda+1}{\lambda+\gamma}\right) \le \alpha_\lambda^{\lceil \log_M(d+1) \rceil}$. Condition 2 holds. ◀

3.3 Heuristics behind $\Phi_2(x)$

The most intricate part of our proofs of Theorem 17 and Theorem 18 is the choice of the potential function $\Phi_2(x)$ given by (5). Here we give a brief heuristic of deriving it. It is more of an “educated guess” than a rigorous argument.

We want to pick $\Phi_2(x)$ such that Condition 2 holds. In particular, we want

$$\varphi_2(F_d(\mathbf{x})) \sum_{i=1}^{d} \frac{\partial F_d}{\partial x_i} \cdot \frac{1}{\varphi_2(x_i)} < 1.$$

It is fair to assume that the left hand side of the equation above takes its maximum when all x_i's are equal. Hence, we hope the following to hold

$$\frac{\varphi_2(f_d(x)) f_d'(x)}{\varphi_2(x)} < 1, \tag{8}$$

where $f_d(x) = \lambda \left(\frac{\beta x+1}{x+\gamma}\right)^d$ is the symmetrized version of $F_d(\mathbf{x})$. We will use $z := f_d(x)$ to simplify notation. Since we want (8) to hold for all degrees d, we hope to eliminate d from the left hand side of (8). Notice that $\varphi_2(x)$ should be independent from d. Therefore, we take the derivative of $\varphi_2(f_d(x))f'_d(x)$ against d and get

$$\begin{aligned}\frac{\partial \varphi_2(f_d(x))f'_d(x)}{\partial d} &= \frac{\beta\gamma - 1}{(\beta x+1)(x+\gamma)}\left(\varphi_2(z)z + \varphi_2(z)z\log\frac{z}{\lambda} + \varphi'_2(z)z^2\log\frac{z}{\lambda}\right)\\ &= \frac{(\beta\gamma-1)z\varphi_2(z)}{(\beta x+1)(x+\gamma)}\left(1+\log\frac{z}{\lambda} + (\log\varphi_2(z))'z\log\frac{z}{\lambda}\right).\end{aligned}$$

We may achieve our goal of eliminating d by imposing the sum in the last parenthesis to be 0, namely

$$\begin{aligned}(\log\varphi_2(z))' &= -\frac{1}{z} - \frac{1}{z\log\frac{z}{\lambda}}\\ &= -(\log z)' - \left(\log\log\frac{\lambda}{z}\right)'. \end{aligned} \tag{9}$$

From (9), it is easy to see that $\varphi_2(z) = \frac{1}{z\log\frac{\lambda}{z}}$ satisfies our need. To get the full definition of (5), we apply a thresholding trick to bound $\varphi_2(z)$ away from 0.

3.4 Discussion of the $\beta > 1$case

We cannot combine conditions of Theorem 17 and Theorem 18 together to have an FPTAS. In particular, when $\beta > 1$ strong special mixing fails for any λ even if $\lambda < \lambda_c$. To see this, given a Δ-ary tree T, we can append t many children to every vertex in T to get a new tree T' and impose a partial configuration σ where all these new children are pinned to 0. Effectively, the tree T' is equivalent to T where every vertex has a new external field of $\lambda\beta^t$, which is larger than λ_c^{int} if t is sufficiently large regardless of λ. Then by Proposition 6, long range correlation exists in T' with the partial configuration σ, and strong spatial mixing fails.

On the other hand, it is easy to see from the proof that, Theorem 17 can be generalized to allow a partial configuration σ on some subset Λ where the marginal probability of every vertex $v \in \Lambda$ satisfies $p_v^\sigma \le \frac{\lambda_c}{\lambda_c+1}$. This is not the case for the SAW tree which our algorithm relies on when $\beta > 1$. However, the following observation shows that if $\lambda_v \le \lambda_c \le \frac{\gamma-1}{\beta-1}$, then the marginal probability of any instance G satisfies this requirement. Thus, it seems the only piece missing to obtain an algorithm is to design a better recursion tree instead of the SAW tree.

▶ **Proposition 20.** *Let (β,γ) be two parameters such that $1 \le \beta \le \gamma$ and $\beta\gamma > 1$. Let $\lambda \le \frac{\gamma-1}{\beta-1}$ be another parameter. For any graph $G = (V,E)$, if $\pi(v) \le \lambda$ for all $v \in V$, then $p_v \le \frac{\lambda}{\lambda+1}$.*

To prove this proposition, we need to use the random cluster formulation of 2-spin models. Let G be a graph and $e = (v_1, v_2)$ be one of its edges. Let G^+ be the graph where the edge e is contracted, and G^- be the graph where e is removed. Moreover, in G^+, we assign $\pi^+(\widetilde{v}) = \lambda_{v_1}\lambda_{v_2}\frac{\beta-1}{\gamma-1}$, where $\widetilde{v}$ is the vertex obtained from contacting e. Then we have that

$$Z(G) = Z(G^-) + (\gamma-1)Z(G^+), \tag{10}$$

where we write $Z(G)$ instead of $Z_{\beta,\gamma,\pi}(G)$ to simplify the notation. To show the equation above we only need a simple adapation of the random cluster formulation of the Ising model to the 2-spin setting.

Proof of Proposition 20. Suppose $G = (V, E)$ where $|V| = n$ and $|E| = m$. We show the claim by inducting on (m, n). Clearly the statement holds when $m = 0$ or $n = 1$. Hence we may assume the claim holds for (m', n) where $m' < m$ as well as (m', n') where $n' < n$, and show that the claim holds for (m, n).

Pick an arbitrary edge $e = (v_1, v_2)$ in G. Let G^+ and G^- be as in the random cluster formulation. It is easy to see that $\pi(\widetilde{v}) = \lambda_{v_1}\lambda_{v_2}\frac{\beta-1}{\gamma-1} \le \lambda$. Hence both G^+ and G^- satisfy the induction hypothesis. It implies that $p_{G^-;v} \le \frac{\lambda}{\lambda+1}$ for any v, where $p_{G^-;v}$ is the mariginal probability of v in G^-. Moreover, $p_{G^+;v} \le \frac{\lambda}{\lambda+1}$ for any $v \in V^+$, where V^+ is the vertex set of G^+. Let δ be a mapping $V \to V^+$ such that $\delta(v) = v$ if $v \ne v_1, v_2$ and $\delta(v_1) = \delta(v_2) = \widetilde{v}$. Then using (10) we have that for any vertex $v \in V$,

$$\begin{aligned} p_{G;v} &= \frac{Z^{\sigma(v)=0}(G)}{Z(G)} = \frac{Z^{\sigma(v)=0}(G^-) + (\gamma-1)Z^{\sigma(\delta(v))=0}(G^+)}{Z(G^-) + (\gamma-1)Z(G^+)} \\ &= p_{G^-;v} \cdot \frac{Z(G^-)}{Z(G^-) + (\gamma-1)Z(G^+)} + p_{G^+;\delta(v)} \cdot \frac{(\gamma-1)Z(G^+)}{Z(G^-) + (\gamma-1)Z(G^+)} \\ &\le \frac{\lambda}{\lambda+1} \cdot \frac{Z(G^-)}{Z(G^-) + Z(G^+)} + \frac{\lambda}{\lambda+1} \cdot \frac{(\gamma-1)Z(G^+)}{Z(G^-) + (\gamma-1)Z(G^+)} = \frac{\lambda}{\lambda+1}, \end{aligned}$$

where in the last line we use the induction hypotheses. ◀

4 Correlation Decay Beyond λ_c

Let β, γ be two parameters such that $\beta \le 1 < \gamma$ and $\beta\gamma > 1$. In this section we give an example to show that if Δ_c is not an integer, then correlation decay still holds for a small interval beyond λ_c. To simplify the presentation, we assume that π is a uniform field such that $\pi(v) = \lambda$. Note that the potential function $\varphi_2(x)$ does not extend beyond λ_c.

Let $\beta = 0.6$ and $\gamma = 2$. Then $\Delta_c = \frac{\sqrt{\beta\gamma}+1}{\sqrt{\beta\gamma}-1} \approx 21.95$ and $\lambda_c = (\gamma/\beta)^{\frac{\Delta_c+1}{2}} < 1002761$. Let $\lambda = 1002762 > \lambda_c$. We will show that #2SPIN(β, γ, λ) still has an FPTAS.

Define a constant t as

$$t := \frac{\sqrt{\beta\gamma}+1}{\sqrt{\beta\gamma}-1} \cdot \frac{\log\sqrt{\gamma/\beta}}{\sqrt{\gamma/\beta}+1} - \log\left(1 + \sqrt{\beta/\gamma}\right) \approx 4.24032. \tag{11}$$

We consider the potential function $\Phi_3(x)$ so that $\varphi_3(x) := \frac{1}{x(\log(1+1/x)+t)}$. With this choice,

$$\begin{aligned} C_{\varphi_3,d}(\mathbf{x}) &= \varphi_3(F_d(\mathbf{x})) \sum_{i=1}^{d} \frac{\partial F_d}{\partial x_i} \cdot \frac{1}{\varphi_3(x)} \\ &= \frac{\beta\gamma - 1}{\log(1 + 1/F_d(\mathbf{x})) + t} \sum_{i=1}^{d} \frac{x_i(\log(1+1/x_i)+t)}{(\beta x_i+1)(x_i+\gamma)}. \end{aligned}$$

We do a change of variables. Let $r_i = \frac{\beta x_i+1}{x_i+\gamma}$. Then $x_i = \frac{\gamma r_i - 1}{\beta - r_i}$, $\beta x_i + 1 = \frac{r_i(\beta\gamma-1)}{\beta-r_i}$, and $x_i + \gamma = \frac{\beta\gamma-1}{\beta-r_i}$. Hence,

$$\begin{aligned} \sum_{i=1}^{d} \frac{x_i(\log(1+1/x_i)+t)}{(\beta x_i+1)(x_i+\gamma)} &= \sum_{i=1}^{d} \frac{(\gamma r_i - 1)(\beta - r_i)}{r_i(\beta\gamma-1)^2} \cdot \left(\log\left(1 + \frac{\beta - r_i}{\gamma r_i - 1}\right) + t\right) \\ &= \frac{1}{(\beta\gamma-1)^2} \sum_{i=1}^{d} \left(1 + \beta\gamma - \frac{\beta}{r_i} - \gamma r_i\right)\left(\log\left(1 + \frac{\beta - r_i}{\gamma r_i - 1}\right) + t\right). \end{aligned}$$

Furthermore, let $s_i = \log r_i$. As $r_i \in \left(\frac{1}{\gamma}, \beta\right)$, $s_i \in (-\log\gamma, \log\beta)$. Let

$$\rho(x) := \left(1 + \beta\gamma - \beta e^{-x} - \gamma e^{x}\right)\left(\log\left(1 + \frac{\beta - e^{x}}{\gamma e^{x} - 1}\right) + t\right).$$

Then $\rho(x)$ is concave for any $x \in (-\log\gamma, \log\beta)$. It can be easily verified, as the second derivative is

$$\begin{aligned}\rho''(x) &= \frac{(\beta+1)(\beta\gamma-1)}{\beta - 1 + e^{x}(\gamma-1)} + \frac{\gamma(\beta\gamma-1)}{\gamma-1} - \frac{\beta(\beta\gamma-1)}{\beta - e^{x}} - \frac{(\beta-1)(\beta\gamma-1)^2}{(\gamma-1)(\beta-1+e^{x}(\gamma-1))^2} \\ &\quad - \beta t e^{-x} - \gamma t e^{x} - e^{-x}\left(\beta + e^{2x}\gamma\right) Log\left(1 + \frac{\gamma e^{x} - 1}{\beta - e^{x}}\right). \\ &\le \gamma(\beta+1) + \frac{\gamma(\beta\gamma-1)}{\gamma-1} - \beta\gamma - \frac{\beta-1}{\gamma-1} - 2t < -5.68 < 0, \end{aligned} \tag{12}$$

where in the last line we used (11) and the fact that $1/\gamma \le e^{x} \le \beta$. Hence, by concavity, we have that for any $x_i \in (0, \lambda]$,

$$\begin{aligned} C_{\varphi_3,d}(\mathbf{x}) &= \frac{\beta\gamma - 1}{\log\left(1 + 1/F_d(\mathbf{x})\right) + t} \sum_{i=1}^{d} \frac{x_i\left(\log(1 + 1/x_i) + t\right)}{(\beta x_i + 1)(x_i + \gamma)}, \\ &\le \frac{\beta\gamma - 1}{\log\left(1 + 1/f_d(\widetilde{x})\right) + t} \cdot \frac{d\widetilde{x}\left(\log(1 + \widetilde{x}^{-1}) + t\right)}{(\beta\widetilde{x} + 1)(\widetilde{x} + \gamma)} = c_{\varphi_3,d}(\widetilde{x}), \end{aligned} \tag{13}$$

where $\widetilde{x} > 0$ is the unique solution such that $f_d(\widetilde{x}) = F_d(\mathbf{x})$.

Next we show that there exists an $\alpha < 1$ such that for any integer d and $x > 0$, $c_{\varphi_3,d}(x) < \alpha$. In fact, by (11), our choice of t, it is not hard to show that the maximum of $c_{\varphi_3,d}(x)$ is achieved at $x = \sqrt{\gamma/\beta}$ and $d = \Delta_c$, which is 1 if $\lambda = \lambda_c$ and is larger than 1 if $\lambda > \lambda_c$. However, since the degree d has to be an integer, we can verify that for any integer $1 \le d \le 100$, the maximum of $c_{\varphi_3,d}(x)$ is $c_{\varphi_3,22}(x_{22}) = 0.999983$ where $x_{22} \approx 1.83066$. If $d > 100$, then

$$\begin{aligned} c_{\varphi_3,d}(x) &= \frac{d(\beta\gamma - 1)}{\log\left(1 + 1/f_d(x)\right) + t} \cdot \frac{x\left(\log(1 + x^{-1}) + t\right)}{(\beta x + 1)(x + \gamma)} \\ &\le C_0 \cdot C_1 < 1, \end{aligned}$$

where $C_0 < 1.07191$ is the maximum of $\frac{x\left(\log(1+x^{-1})+t\right)}{(\beta x+1)(x+\gamma)}$ for any $x > 0$, and $C_1 < 0.481875$ is the maximum of $\frac{d(\beta\gamma-1)}{\log(1+\lambda^{-1}\beta^{-d})+t}$ for any $d > 100$. Then, due to (13), we have that for any $x_i \in (0, \lambda]$, $C_{\varphi_3,d}(\mathbf{x}) < \alpha = 0.999983 < 1$. This is the counterpart of $C_{\varphi_2,d}(\mathbf{x}) < \alpha_\lambda$ in the proof of Theorem 18. To make $\varphi_3(x)$ satisfy Condition 1 and Condition 2 in Definition 10, it is sufficient to do a simple "chop-off" trick to $\varphi_3(x)$ as in (5). We will omit the detail here.

▶ **Proposition 21.** *For $\beta = 0.6$, $\gamma = 2$, and $\lambda = 1002762 > \lambda_c$, #2*Spin*$(\beta, \gamma, \lambda)$ has an FPTAS.*

It is easy to see that the argument above works for any $\beta \le 1 < \gamma$ and $\beta\gamma > 1$ except (12), the concavity of $\rho(x)$. Indeed, the concavity does not hold if, say, $\beta = 1$ and $\gamma = 2$. Nevertheless, the key point here is that λ_c is not the tight bound for FPTAS. Short of a conjectured optimal bound, we did not try to optimize the potential function nor the applicable range of the proof above.

5 Limitations of Correlation Decay

In this section, we discuss some limitations of approximation algorithms for ferromagnetic 2-spin models based on correlation decay analysis.

The problem of counting independent sets in bipartite graphs (#BIS) plays an important role in classifying approximate counting complexity. #BIS is not known to have any efficient approximation algorithm, despite many attempts. However there is no known approximation preserving reduction (AP-reduction) to reduce #BIS from #SAT either. It is conjectured to have intermediate approximation complexity, and in particular, to have no FPRAS [3].

Goldberg and Jerrum [6] showed that for any $\beta\gamma > 1$, approximating $\#2\text{SPIN}(\beta,\gamma,(0,\infty))$ can be reduced to approximating #BIS. This is the (approximation) complexity upper bound of all ferromagnetic 2-spin models. In contrast, by Theorem 13, $\#\Delta\text{-}2\text{SPIN}(\beta,\gamma,(0,\infty))$ has an FPTAS, if $\Delta < \Delta_c + 1$. Note that when we write $\#2\text{SPIN}(\beta,\gamma,(0,\infty))$ the field is implicitly assumed to be at most polynomial in size of the graph (or in unary).

We then consider fields with some constant bounds. Recall that $\lambda_c^{int} = (\gamma/\beta)^{\frac{\lceil\Delta_c\rceil+1}{2}}$. Let $\lambda_c^{int'} = (\gamma/\beta)^{\frac{\lfloor\Delta_c\rfloor+2}{2}}$. Then $\lambda_c^{int'} = \lambda_c^{int}$ unless Δ_c is an integer. By reducing to anti-ferromagnetic 2-spin models in bipartite graphs, we have the following hardness result, which is first observed in [13, Theorem 3].

▶ **Proposition 22.** *Let (β,γ,λ) be a set of parameters such that $\beta < \gamma$, $\beta\gamma > 1$, and $\lambda > \lambda_c^{int'}$. Then $\#2\text{SPIN}(\beta,\gamma,(0,\lambda])$ is #BIS-hard.*

The reduction goes as follows. Anti-ferromagnetic Ising models with a constant non-trivial field in bounded degree bipartite graphs are #BIS-hard, if the uniqueness condition fails [1]. Given such an instance, we may first flip the truth table of one side. This effectively results in a ferromagnetic Ising model in the same bipartite graph, with two different fields on each side. By a standard diagonal transformation, we can transform such an Ising model to any ferromagnetic 2-spin model, with various local fields depending on the degree. It can be verified that for any $\lambda > \lambda_c^{int'}$, we may pick a field in the anti-ferromagnetic Ising model to start with, such that uniqueness fails and after the transformation, the largest field in use is at most λ.

The hardness bound in Proposition 22 matches the failure of uniqueness due to Proposition 6, unless Δ_c is an integer. In contrast to Proposition 22, Theorem 18 implies that if $\beta \le 1 < \gamma$ and $\lambda < \lambda_c = (\gamma/\beta)^{\frac{\Delta_c+1}{2}}$, then $\#2\text{SPIN}(\beta,\gamma,(0,\lambda])$ has an FPTAS. Hence Theorem 18 is almost optimal, up to an integrality gap.

We note that λ_c is not the tight bound for FPTAS, as observed in Proposition 21. Since the degree d has to be an integer, with an appropriate choice of the potential function, there is a small interval beyond λ_c such that strong spatial mixing still holds. Interestingly, it seems that λ_c^{int} is not the right bound either. Let us make a concrete example. Let $\beta = 1$ and $\gamma = 2$. Then $\Delta_c = \frac{\sqrt{\beta\gamma}+1}{\sqrt{\beta\gamma}-1} = \frac{\sqrt{2}+1}{\sqrt{2}-1} \approx 5.82843$. Hence $\lambda_c \approx 10.6606$ and $\lambda_c^{int} = (2)^{\frac{6+1}{2}} \approx 11.3137$. However, even if $\lambda < \lambda_c^{int}$, the system may not exhibit spatial mixing, neither in the strong nor in the weak sense.

In fact, even the spatial mixing in the sense of Theorem 1 does not necessarily hold if $\lambda < \lambda_c^{int}$. To see this, we take any $\lambda \in [10.9759, 10.9965]$ so that $\lambda_c < \lambda < \lambda_c^{int}$. Consider an infinite tree where at even layers, each vertex has 5 children, and at odd layers, each vertex has 7 children. There are more than one Gibbs measures in this tree. This can be easily verified from the fact that the two layer recursion function $f_5(f_7(x))$ has three fixed points such that $x = f_5(f_7(x))$. In addition, all three fixed points $\widehat{x}_i$ satisfy that $\widehat{x}_i < \lambda_c$ for $i = 1, 2, 3$. Consider a tree T with alternating degrees 5 and 7 of depth 2ℓ, and another tree T' of the same structure in the first 2ℓ layers as T but with one more layer where each vertex has, say, 50 children. It is not hard to verify that as ℓ increases, the marginal ratio at the root of T converges to $\widehat{x}_3$, but the ratio at the root of T' converges to $\widehat{x}_1$. This example indicates that one should not expect correlation decay algorithms to work all the way up to λ_c^{int}.

At last, if we consider the uniform field case #2SPIN(β, γ, λ), then our tractability results still holds. However, to extend the hardness results as in Proposition 22 from an interval of fields to a uniform one, there seems to be some technical difficulty. Suppose we want to construct a combinatorial gadget to effectively realize another field. There is a gap between λ and the next largest possible field to realize. This is why in [13], there are some extra conditions transiting from an interval of fields to the uniform case. The observation above about the failure of SSM in irregular trees may suggest a random bipartite construction of uneven degrees. However, to analyze such a gadget is beyond the scope of the current paper.

6 Missing Proofs

At last, we gather technical details and proofs that are omitted in Section 2.2, Section 2.3, and Section 3.2.

6.1 Details about the Uniqueness Threshold

We want to prove Propositions 5 and Proposition 6. Technically by only considering the symmetric recursion $f_d(x) = \lambda \left(\frac{\beta x+1}{x+\gamma}\right)^d$, we are implicitly assuming uniform boundary conditions. If there are more than one fixed points for $f_d(x)$, then clearly there are multiple Gibbs measures. Hence, $f_d(x)$ having only one fixed point is a necessary condition for the uniqueness condition in $\mathbb{T}_{d+1}$. Moreover, it is also sufficient. The reason is that the influence on the root of an arbitrary boundary condition is bounded between those of the all "0" and all "1" boundary conditions.

First do some calculation here. Take the derivative of $f_d(x)$:

$$f_d'(x) = \frac{d(\beta\gamma - 1) f_d(x)}{(\beta x + 1)(x + \gamma)}. \tag{14}$$

Then take the second derivative:

$$f_d''(x) = \frac{f_d'(x)}{f_d(x)} - \frac{\beta}{\beta x + 1} - \frac{1}{x + \gamma} = \frac{d(\beta\gamma - 1) - \beta\gamma - 1 - 2\beta x}{(\beta x + 1)(x + \gamma)}.$$

Therefore, at $x^* := \frac{d(\beta\gamma-1)-(\beta\gamma+1)}{2\beta}$, $f_d''(x^*) = 0$. It's easy to see when $d < \frac{\beta\gamma+1}{\beta\gamma-1}$, $f_d''(x) < 0$ for all $x > 0$. So $f_d(x)$ is concave and therefore has only one fixed point.

Since $f_d(x)$ has only one inflection point, there are at most three fixed points. Moreover, the uniqueness condition is equivalent to say that for all fixed points $\widehat{x}_d$ of $f_d(x)$, $f_d'(\widehat{x}_d) < 1$. For a fixed point $\widehat{x}_d$, we plug it in (14):

$$f_d'(\widehat{x}_d) = \frac{d(\beta\gamma - 1)\widehat{x}_d}{(\beta\widehat{x}_d + 1)(\widehat{x}_d + \gamma)}.$$

Recall that $\Delta_c := \frac{\sqrt{\beta\gamma}+1}{\sqrt{\beta\gamma}-1}$. If $d < \Delta_c$, we have that for any x,

$$\begin{aligned}(\beta x + 1)(x + \gamma) - d(\beta\gamma - 1)x &= \beta x^2 + ((\beta\gamma + 1) - d(\beta\gamma - 1))x + \gamma \\ &> \beta x^2 + (\beta\gamma + 1 - (\sqrt{\beta\gamma} + 1)^2)x + \gamma \\ &= (\sqrt{\beta}x - \sqrt{\gamma})^2 \geq 0.\end{aligned}$$

Hence $(\beta x+1)(x+\gamma) > d(\beta\gamma-1)x$. In particular, $f_d'(\widehat{x}_d) < 1$ for any fixed point $\widehat{x}_d$ and the uniqueness condition holds. This proves Proposition 5.

To show Proposition 6, we may assume that $d \geq \Delta_c$. We may also assume that $\beta \leq \gamma$. The equation $(\beta x+1)(\gamma+x) = d(\beta\gamma-1)x$ has two solutions, which are

$$x_0 = x^* - \frac{\sqrt{((\beta\gamma+1)-d(\beta\gamma-1))^2-4\beta\gamma}}{2\beta}$$
$$\text{and} \quad x_1 = x^* + \frac{\sqrt{((\beta\gamma+1)-d(\beta\gamma-1))^2-4\beta\gamma}}{2\beta}.$$

Notice that both of them are positive since $x_0+x_1 = 2x^* > 0$ and $x_0x_1 = \beta/\gamma$.

We show that $f_d(x_0) > x_0$ or $f_d(x_1) < x_1$ is equivalent to the uniqueness condition. First we assume this condition doesn't hold, that is $f_d(x_0) \leq x_0$ and $f_d(x_1) \geq x_1$. If any of the equation holds, then x_0 or x_1 is a fixed point and the derivative is 1. So we have non-uniqueness. Otherwise, we have $f_d(x_0) < x_0$ and $f_d(x_1) > x_1$. Since $x_0 < x_1$, there is some fixed point $\widetilde{x}$ satisfying $f_d(\widetilde{x}) = \widetilde{x}$ and $x_0 < \widetilde{x} < x_1$. The second inequality implies that $(\beta\widetilde{x}+1)(\widetilde{x}+\gamma) < d(\beta\gamma-1)\widetilde{x}$. Therefore $f_d'(\widetilde{x}) > 1$ and non-uniqueness holds.

To show the other direction, if $f_d(x_0) > x_0$, then

$$f_d'(x_0) = \frac{d(\beta\gamma-1)f(x_0)}{(\beta x_0+1)(x_0+\gamma)} > \frac{d(\beta\gamma-1)x_0}{(\beta x_0+1)(x_0+\gamma)} = 1.$$

Assume for contradiction that $f_d(x)$ has three fixed points, denoted by $\widetilde{x}_0 < \widetilde{x}_1 < \widetilde{x}_2$. Then the middle fixed point $\widetilde{x}_1$ satisfies $f_d'(\widetilde{x}_1) > 1$. Therefore $\widetilde{x}_1 > x_0$ and there are two fixed points larger than x_0. However, for $x_0 < x \leq x^*$, $f_d'(x) > 1$ and $f_d(x_0) > x_0$. Hence there is no fixed point in this interval. For $x > x^*$, the function is concave and has exactly one fixed point. So there is only 1 fixed point larger than x_0. Contradiction. The case that $f_d(x_1) < x_1$ is similar.

These two conditions could be rewritten as

$$\lambda > \frac{x_0(x_0+\gamma)^d}{(\beta x_0+1)^d} \tag{15}$$

and

$$\lambda < \frac{x_1(x_1+\gamma)^d}{(\beta x_1+1)^d}. \tag{16}$$

Notice that the right hand side has nothing to do with λ in both (15) and (16).

We want to see how conditions (15) and (16) change as d changes. Treat d as a continuous variable. Define

$$g_i(d) := \frac{x_i(x_i+\gamma)^d}{(\beta x_i+1)^d}.$$

where $i = 0, 1$ and x_i is defined above depending on β, γ and d. Take the derivative:

$$\begin{aligned}
\frac{g_i'(d)}{g_i(d)} &= \frac{\partial x_i}{\partial d}\left(\frac{1}{x_i} + \frac{d}{x_i+\gamma} - \frac{d\beta}{\beta x_i+1}\right) + \log(x_i+\gamma) - \log(\beta x_i+1) \\
&= \frac{\partial x_i}{\partial d}\left(\frac{1}{x_i} + \frac{d(1-\beta\gamma)}{(x_i+\gamma)(\beta x_i+1)}\right) + \log\frac{x_i+\gamma}{\beta x_i+1} \\
&= \frac{\partial x_i}{\partial d}\left(\frac{1}{x_i} - \frac{1}{x_i}\right) + \log\frac{x_i+\gamma}{\beta x_i+1} = \log\frac{x_i+\gamma}{\beta x_i+1}.
\end{aligned}$$

If $\beta \le 1$ these two functions are increasing in d. Recall that $\Delta_c = \frac{\sqrt{\beta\gamma}+1}{\sqrt{\beta\gamma}-1}$, and $\lambda_c^{int} = g_1(\lceil\Delta_c\rceil) = (\gamma/\beta)^{\frac{\lceil\Delta_c+1\rceil}{2}}$. Thus if $\lambda < \lambda_c^{int}$, (16) holds for all integers d. On the other hand,

$$\begin{aligned} g_0(d) &= \frac{x_0(x_0+\gamma)^d}{(\beta x_0+1)^d} > x_0\beta^{-d} = \frac{\beta}{\gamma x_1}\beta^{-d} > \frac{\beta}{\gamma 2x^*}\beta^{-d} \\ &= \frac{\beta^2}{\gamma(d(\beta\gamma-1)-(\beta\gamma+1))}\cdot\beta^{-d} \\ &\to \infty \text{ as } d \text{ goes to } \infty. \end{aligned}$$

Hence there is no λ such that (15) holds for all integers d. This proves Proposition 6.

If $\beta > 1$, then neither (15) nor (16) can hold for all integers d. The reason is

$$\begin{aligned} g_0(d) &= \frac{x_0(x_0+\gamma)^d}{(\beta x_0+1)^d} = \frac{x_0(x_0+\gamma)^{2d}}{(d(\beta\gamma-1)x_0)^d} > x_0\left(\frac{\gamma}{d(\beta\gamma-1)x_0}\right)^d \\ &\to \infty \text{ as } d \text{ goes to } \infty, \end{aligned}$$

as $d(\beta\gamma-1)x_0 < \gamma$ for sufficiently large d, and

$$\begin{aligned} g_1(d) &= \frac{x_1(x_1+\gamma)^d}{(\beta x_1+1)^d} = \frac{x_1(d(\beta\gamma-1)x_1)^d}{(\beta x_1+1)^{2d}} < x_1\left(\frac{d(\beta\gamma-1)}{\beta^2 x_1}\right)^d \\ &\to 0 \text{ as } d \text{ goes to } \infty, \end{aligned}$$

as $\beta^2 x_1 > d(\beta\gamma-1)$ for sufficiently large d.

6.2 Details about the Potential Method

In this section we provide missing details and proofs in Section 2.3.

To study correlation decay on trees, we use the standard recursion given in (2). Recall that T is a tree with root v. Vertices $v_1,\dots,v_d$ are d children of v, and T_i is the subtree rooted by v_i. A configuration σ_Λ is on a subset Λ of vertices, and R_T^σ denote the ratio of marginal probabilities at v given a partial configuration σ on T.

We want to study the influence of another set of vertices, say S, upon v. In particular, we want to study the range of ratios at v over all possible configurations on S. To this end, we define the lower and upper bounds as follows. Notice that as S will be fixed, we may assume that it is a subset of Λ.

▶ **Definition 23.** Let $T, v, \Lambda, \sigma_\Lambda, S, R_T^\sigma$ be as above. Define $R_v := \min_{\tau_\Lambda} R_T^{\tau_\Lambda}$ and $R^v := \max_{\tau_\Lambda} R_T^{\tau_\Lambda}$, where τ_Λ can only differ from σ_Λ on S. Define $\delta_v := R^v - R_v$.

Our goal is thus to prove that $\delta_v \le \exp(-\Omega(\text{dist}(v,S)))$. We can recursively calculate R_v and R^v as follows. The base cases are:

1. $v \in S$, in which case $R_v = 0$ and $R^v = \infty$ and $\delta_v = \infty$;
2. $v \in \Lambda \setminus S$, i.e. v is fixed to be the same value in all τ_Λ, in which case $R_v = R^v = 0$ (or ∞) if v is fixed to be blue (or green), and $\delta_v = 0$;
3. $v \notin \Lambda$ and v is the only node of T, in which case $R_v = R^v = \lambda$ and $\delta_v = 0$.

For $v \notin \Lambda$, since F_d is monotonically increasing with respect to any x_i for any $\beta\gamma > 1$,

$$R_v = F_d(R_{v_1},\dots,R_{v_d}) \text{ and } R^v = F_d(R^{v_1},\dots,R^{v_d}),$$

where R_{v_i} and R^{v_i} are recursively defined lower and upper bounds of $R_{T_i}^{\tau_\Lambda}$ for $1 \le i \le d$.

$$\begin{tikzcd}
\mathbf{x} \arrow[r, shift left, harpoon, "\Phi"] \arrow[d, "F_d"'] & \mathbf{y} \arrow[l, shift left, harpoon, "\Phi^{-1}"] \arrow[d, dashed, "G_d"'] \\
F_d(\mathbf{x}) \arrow[r, shift left, harpoon, "\Phi"] & G_d(\mathbf{y}) \arrow[l, shift left, harpoon, "\Phi^{-1}"]
\end{tikzcd}$$

Figure 1 Commutative diagram between F_d and G_d.

Our goal is to show that δ_v decays exponentially in the depth of the recursion under certain conditions such as the uniqueness. A straightforward approach would be to prove that δ_v contracts by a constant ratio at each recursion step. This is a sufficient, but not necessary condition for the exponential decay. Indeed there are circumstances that δ_v does not necessarily decay in every step but does decay in the long run. To amortize this behaviour, we use a *potential function* $\Phi(x)$ and show that the correlation of a new recursion decays by a constant ratio.

To be more precise, the potential function $\Phi : \mathbb{R}^+ \to \mathbb{R}^+$ is a differentiable and monotonically increasing function. It maps the domain of the original recursion to a new one. Let $y_i = \Phi(x_i)$. We want to consider the recursion for y_i's. The new recursion function, which is the pullback of F_d, is defined as

$$G_d(y_1, \ldots, y_d) := \Phi(F_d(\Phi^{-1}(x_1), \ldots, \Phi^{-1}(x_d))).$$

The relationship between $F_d(\mathbf{x})$ and $G_d(\mathbf{y})$ is illustrated in Figure 1.

We want to prove Lemma 8 and Lemma 12. To do so, we also define the upper and lower bounds of y. Define $y_v = \Phi(R_v)$ and accordingly $y_{v_i} = \Phi(R_{v_i})$, for $1 \le i \le d$, as well as $y^v = \Phi(R^v)$ and $y^{v_i} = \Phi(R^{v_i})$, for $1 \le i \le d$. We have that

$$y_v = G_d(y_{v_1}, \ldots, y_{v_d}) \text{ and } y^v = G_d(y^{v_1}, \ldots, y^{v_d}). \tag{17}$$

Let $\varepsilon_v = y^v - y_v$. For a good potential function, exponential decay of ε_v is sufficient to imply that of δ_v.

▶ **Lemma 24.** *Let $\Phi(x)$ be a good potential function for the field λ at v. Then there exists a constant C such that $\delta_v \le C\varepsilon_v$ for any* $\operatorname{dist}(v, S) \ge 2$.

Proof. By (17) and the Mean Value Theorem, there exists an $\widetilde{R} \in [R_v, R^v]$ such that

$$\varepsilon_v = \Phi(R^v) - \Phi(R_v) = \Phi'(\widetilde{R}) \cdot \delta_v = \varphi(\widetilde{R}) \cdot \delta_v. \tag{18}$$

Since $\operatorname{dist}(v, S) \ge 2$, we have that $R_v \ge \lambda\gamma^{-d}$ and $R^v \le \lambda\beta^d$. Hence $\widetilde{R} \in [\lambda\gamma^{-d}, \lambda\beta^d]$, and by Condition 1 of Definition 7, there exists a constant C_1 such that $\varphi(\widetilde{R}) \ge C_1$. Therefore $\delta_v \le 1/C_1\varepsilon_v$. ◀

The next lemma explains Condition 2 of Definition 7.

▶ **Lemma 25.** *Let $\Phi(x)$ be a good potential function with contraction ratio α. Then,*

$$\varepsilon_v \le \alpha \max_{1 \le i \le d} \{\varepsilon_{v_i}\}.$$

Proof. First we use (17):

$$\varepsilon_v = y^v - y_v = G_d(y^{v_1}, \dots, y^{v_d}) - G_d(y_{v_1}, \dots, y_{v_d}).$$

Let $\mathbf{y}_1 = (y^{v_1}, \dots, y^{v_d})$ and $\mathbf{y}_0 = (y_{v_1}, \dots, y_{v_d})$. Let $\mathbf{z}(t) = t\mathbf{y}_1 + (1-t)\mathbf{y}_0$ be a linear combination of $\mathbf{y}_0$ and $\mathbf{y}_1$ where $t \in [0,1]$. Then we have that

$$\varepsilon_v = G_d(\mathbf{z}(1)) - G_d(\mathbf{z}(0)).$$

By the Mean Value Theorem, there exist $\widetilde{t}$ such that $\varepsilon_v = \frac{\mathrm{d}\, G_d(\mathbf{z}(t))}{\mathrm{d}\, t}\Big|_{t=\widetilde{t}}$. Let $\widetilde{y_i} = \widetilde{t}y^{v_i} + (1-\widetilde{t})y_{v_i}$ for all $1 \le i \le d$. Then we have that

$$\varepsilon_v = \left|\nabla G_d(\widetilde{y_1}, \dots, \widetilde{y_d}) \cdot (\varepsilon_{v_1}, \dots, \varepsilon_{v_d})\right|. \tag{19}$$

It is straightforward to calculate that

$$\frac{\partial G_d(\mathbf{y})}{\partial y_i} = \frac{\varphi(F_d(\mathbf{R}))}{\varphi(R_i)} \cdot \frac{\partial F_d(\mathbf{R})}{\partial R_i}, \tag{20}$$

where $R_i = \Phi^{-1}(y_i)$ and $\mathbf{y}$ and $\mathbf{R}$ are vectors composed by y_i's and R_i's. Plugging (20) into (19) we get that

$$\begin{aligned}\varepsilon_v &= \varphi(F_d(\widetilde{\mathbf{R}})) \cdot \sum_{i=1}^{d} \left|\frac{\partial F_d}{\partial R_i}\right| \frac{1}{\varphi(\widetilde{R_i})} \cdot \varepsilon_{v_i} \\ &\le C_{\varphi,d}(\widetilde{R}_1, \dots, \widetilde{R}_d) \cdot \max_{1 \le i \le d}\{\varepsilon_{v_i}\} \le \alpha \max_{1 \le i \le d}\{\varepsilon_{v_i}\},\end{aligned}$$

where $\widetilde{R_i} = \Phi^{-1}(\widetilde{y_i})$, $\widetilde{\mathbf{R}}$ is the vector composed by $\widetilde{R_i}$'s, and in the last line we use Condition 2 of Definition 7. ◀

Note that the two conditions of a good potential function does not necessarily deal with all cases in the tree recursion. At the root we have one more child than other vertices in a SAW tree. Also, if v has a child $u \in S$, then $\varepsilon_u = \infty$ and the range in both conditions of Definition 7 does not apply. To bound the recursion at the root, we have the following straightforward bound of the original recursion.

▶ **Lemma 26.** *Let (β, γ) be two parameters such that $\beta\gamma > 1$ and $\beta < \gamma$. Let v be a vertex and v_i be its children for $1 \le i \le d$. Suppose $\delta_{v_i} \le C$ for some $C > 0$ and all $1 \le i \le d$. Then,*

$$\delta_v \le d\lambda_v(\beta\gamma - 1)\gamma^{-1}\beta^d C.$$

Proof. It is easy to see that $\gamma \ge 1$. By the same argument as in Lemma 25 and (2), there exists x_i's such that

$$\delta_v = \left|\nabla F_d(x_1, \dots, x_d) \cdot (\delta_{v_1}, \dots, \delta_{v_d})\right| \le C \sum_{i=1}^{d} \left|\frac{\partial F_d(\mathbf{x})}{\partial x_i}\right|,$$

where $\mathbf{x}$ is the vector composed by x_i's. Then, we have that

$$\left|\frac{\partial F_d(\mathbf{x})}{\partial x_i}\right| = \frac{d(\beta\gamma - 1)F_d(\mathbf{x})}{(x_i + \gamma)(\beta x_i + 1)} \le d\lambda_v(\beta\gamma - 1)\gamma^{-1}\beta^d,$$

where we use the fact that $F_d(\mathbf{x}) \le \lambda_v \beta^d$ for any $x_i \in [0, \infty)$ and $\beta\gamma > 1$. The lemma follows. ◀

Now we are ready to prove Lemma 8.

Proof of Lemma 8. Given G and a partial configuration σ_Λ on a subset $\Lambda \subseteq V$ of vertices, we first claim that we can approximate $p_v^{\sigma_\Lambda}$ within additive error ε deterministically in time $O\left(\varepsilon^{\frac{\log \Delta}{\log \alpha}}\right)$. We construct the SAW tree $T = T_{\text{SAW}}(G, v)$. Due to Proposition 3, we only need to approximate $p_v^{\sigma_\Lambda}$ in T, with respect to v and an arbitrary vertex set S. We will also use σ_Λ to denote the configuration in T on Λ_{SAW}. Let S be the set of vertices whose distance to v is larger than t, where t is a parameter that we will specify later. Let δ_v be defined as in Definition 23 with respect to T, v, Λ, σ_Λ, and S. We want to show that $\delta_v = O(\lambda\alpha^t)$.

The maximum degree of T is at most Δ. Thus the root v has at most Δ children in T, and any other vertex in T has at most $\Delta - 1$ children. Assume v has $k \geq 1$ children as otherwise we are done. We may also assume that $v \notin S$ and let $t = \text{dist}(v, S) - 1 \geq 1$. We recursively construct a path $u_0 = v$, u_1,. . . ,u_l of length $l \leq t$ as follows. Given u_i, if there is no child of u_i, then we stop and let $l = i$. Otherwise u_i has at least one child. If $i = t$ then we stop and let $l = t$. Otherwise $l < t$ and let u_{i+1} be the child of u_i such that $\varepsilon_{u_{i+1}}$ takes the maximum ε among all children of u_i. In other words, by Lemma 25, we have that

$$\varepsilon_{u_i} \leq \alpha\varepsilon_{u_{i+1}}, \tag{21}$$

for all $1 \leq i \leq l - 1$. Notice that (21) may not hold for $i = 0$ since $v = u_0$ has possibly Δ children.

First we note that for all $1 \leq i \leq l$, $\text{dist}(v, u_i) = i \leq l \leq t$, and therefore $u_i \notin S$. If we met any vertex u_l with no child, then we claim that $\varepsilon_{u_l} = 0$. This is because u_l is either a free vertex with no child or $u_l \in \Lambda$ but $u_l \notin S$. However since ε_{u_l} takes the maximum ε among all children of u_{l-1}, we have that for all children of u_{i-1}, $\varepsilon = 0$, which implies that $\varepsilon_{u_{i-1}} = 0$. Recursively we get that $\varepsilon_v = \varepsilon_{u_0} = 0$ and clearly the theorem holds by (18).

Hence we may assume that $l = t$. Since $u_l \notin S$, we have that $\delta_{u_l} \leq \lambda_{u_l}\beta^{-(\Delta-1)}$ if $\beta > 1$, or $\delta_{u_l} \leq \lambda_{u_l}$ if $\beta \leq 1$. Hence by (18) and Condition 1 in Definition 7, we have that $\varepsilon_{u_l} \leq C_0$ for some constant C_0. Applying (21) inductively we have that

$$\varepsilon_{u_1} \leq \alpha^l\varepsilon_{u_l} \leq \alpha^t C_0.$$

Hence by Lemma 24, we there exists another constant C_1 such that $\delta_{u_1} \leq \alpha^t C_1$. To get a bound on δ_{u_0}, we use Lemma 26, which states that

$$\delta_{u_0} \leq d_0\lambda_v(\beta\gamma - 1)\gamma^{-1}\beta^{d_0}\delta_{u_1} \leq d_0\lambda_v(\beta\gamma - 1)\gamma^{-1}\beta^{d_0}\alpha^t C_1 = O(\lambda\alpha^t),$$

where $d_0 \leq \Delta$ is the degree of $v = u_0$.

Hence the recursive procedure returns R_v and R^v such that $R_v \leq R_T^{\sigma_\Lambda} \leq R^v$, and $R^v - R_v = O(\lambda\alpha^t)$ where $\alpha < 1$ is the contraction ratio. Note that $R_T^{\sigma_\Lambda} = R_{G,v}^{\sigma_\Lambda} = \frac{p_v^{\sigma_\Lambda}}{1-p_v^{\sigma_\Lambda}}$. Let $p_0 = \frac{R_v}{R_v+1}$ and $p_1 = \frac{R^v}{R^v+1}$. Then $p_0 \leq p_v^{\sigma_\Lambda} \leq p_1$ and

$$p_1 - p_0 = \frac{R^v}{R^v + 1} - \frac{R_v}{R_v + 1} \leq R^v - R_v = O(\lambda\alpha^t). \tag{22}$$

The recursive procedure runs in time $O(\Delta^t)$ since it only needs to construct the first t levels of the self-avoiding walk tree. For any $\varepsilon > 0$, let $t = O(\log_\alpha \varepsilon - \log_\alpha \lambda)$ so that $R^v - R_v < \varepsilon$. This gives an algorithm which approximates $p_v^{\sigma_\Lambda}$ within an additive error ε in time $O\left(\left(\frac{\varepsilon}{\lambda}\right)^{\frac{\log \Delta}{\log \alpha}}\right)$.

Then we use self-reducibility to reduce computing $Z_{\beta,\gamma,\pi}(G)$ to computing conditional marginal probabilities. To be specific, let σ be a configuration on a subset of V and τ be sampled according to the Gibbs measure. Let $p_v^\sigma := \Pr(\tau(v) = 1 \mid \sigma)$ be the conditional marginal probability. We can compute $Z_{\beta,\gamma,\pi}(G)$ from p_v^σ by the following standard procedure. Let $v_1, \dots, v_n$ enumerate vertices in G. For $0 \le i \le n$, let σ_i be the configuration fixing the first i vertices $v_1, \dots, v_i$ as follows: $\sigma_i(v_j) = \sigma_{i-1}(v_j)$ for $1 \le j \le i-1$ and $\sigma_i(v_i)$ is fixed to the spin s so that $p_i := \Pr(\tau(v_i) = s \mid \sigma_{i-1}) \ge 1/3$. This is always possible because clearly

$$\Pr(\tau(v_i) = 0 \mid \sigma_{i-1}) + \Pr(\tau(v_i) = 1 \mid \sigma_{i-1}) = 1.$$

In particular, $\sigma_n \in \{0,1\}^V$ is a configuration of V. The Gibbs measure of σ_n is $\rho(\sigma_n) = \frac{w(\sigma_n)}{Z_{\beta,\gamma,\pi}(G)}$. On the other hand, we can rewrite $\rho(\sigma_n) = p_1 p_2 \cdots p_n$ by conditional probabilities. Thus $Z_{\beta,\gamma,\pi}(G) = \frac{w(\sigma_n)}{p_1 p_2 \cdots p_n}$. The weight $w(\sigma_n)$ given in (1) can be computed exactly in time polynomial in n. Note that p_i equals to either $p_{v_i}^{\sigma_{i-1}}$ or $1 - p_{v_i}^{\sigma_{i-1}}$. Since we can approximate $p_v^{\sigma_\Lambda}$ within an additive error ε in time $O\left(\left(\frac{\varepsilon}{\lambda}\right)^{\frac{\log \Delta}{\log \alpha}}\right)$, the configurations σ_i can be efficiently constructed, which guarantees that all p_i's are bounded away from 0. Thus the product $p_1 p_2 \cdots p_n$ can be approximated within a factor of $(1 \pm n\varepsilon')$ in time $O\left(n\left(\frac{\varepsilon'}{\lambda}\right)^{\frac{\log \Delta}{\log \alpha}}\right)$. Now let $\varepsilon' = \frac{\varepsilon}{n}$. We get the claimed FPTAS for $Z_{\beta,\gamma,\pi}(G)$. ◀

Lemma 11 follows almost immediately from Lemmas 24, 25, and 26 as in the proof above. The only issue is that the range of x should be restricted to $(0, \lambda]$. This is guaranteed by Claim 16.

Finally we show Lemma 12.

Proof of Lemma 12. By the same proof of Lemma 8, we only need to approximate the marginal probability at the root v of a tree T. By Condition 2 of Definition 10, $C_{\varphi,d}(x_1, \cdots, x_d) < \alpha^{\lceil \log_M(d+1) \rceil}$. Denote by $B(\ell)$ the set of all vertices whose M-based depths of v is at most ℓ in T. Hence $|B(\ell)| \le M^\ell$. Let $S = \{u \mid \operatorname{dist}(u, B(\ell)) > 1\}$, which is essentially the same S as in Lemma 8, but under a different metric. We can recursively compute upper and lower bounds R^v and R_v of $R_T^{\sigma_\Lambda}$ such that $R_v \le R_T^{\sigma_\Lambda} \le R^v$, with the base case that for any vertex $u \in S$ trivial bounds $R_u = 0$ and $R^u = \infty$ are used.

We proceed as in the proof of Lemma 8. Without loss of generality, we construct a path $u_0 u_1 \cdots u_k$ in T from the root $u_0 = v$ to a u_k with $\ell_M(u_{k-1}) \le \ell$ and $\ell_M(u_k) > \ell$. As in the proof of Lemma 25, $\varepsilon_{u_j} \le C_{d_j}^{\varphi}(x_{j,1}, \dots, x_{j,d_j}) \cdot \varepsilon_{u_{j+1}}$ for all $0 \le j \le k-1$, where d_j is the number of children of u_j and $x_{j,i} \in [0, \infty)$, $1 \le i \le d_j$. Hence we have that

$$\begin{aligned}\varepsilon_v &\le \varepsilon_{u_k} \cdot \prod_{j=0}^{k-1} \alpha^{\lceil \log_M(d_j+1) \rceil} \le \varepsilon_{u_k} \cdot \alpha^{\sum_{j=0}^{k-1} \lceil \log_M(d_j+1) \rceil} \\ &= \varepsilon_{u_k} \cdot \alpha^{\ell_M(u_k)} \le \varepsilon_{u_k} \cdot \alpha^\ell.\end{aligned}$$

Note that $\operatorname{dist}(u_k, B(\ell)) = 1$ and hence $u_k \notin S$. So $\delta_{u_k} < \lambda_{u_k} \le \lambda$. By (18), we have that $\varepsilon_{u_k} \le \varphi(\widetilde{R})\delta_{u_k}$, for some $\widetilde{R} \in [\lambda_{u_k}\gamma^{-d_k}, \lambda_{u_k}\beta^{d_k}]$. Hence $\varepsilon_{u_k} < C_2\lambda$ by Condition 1 of Definition 10, and $\varepsilon_v < \lambda\alpha^\ell C_2$. By (18) and Condition 1 of Definition 10 again, we have that $\delta_v \le \lambda\alpha^\ell C_2 / C_1$.

The rest of the proof goes the same as that of Lemma 8. The running time has an extra n^2 factor since we need to go down two more levels (in the worst case) outside of $B(\ell)$. ◀

6.3 Proofs of Lemma 14 and Lemma 15

In this section we show Lemma 14 and Lemma 15. We prove Lemma 14 first, and then use it to show Lemma 15.

Proof of Lemma 14. It is trivial if $\beta \le 1$. Now assume that $\beta > 1$. As $\frac{\beta x+1}{x+\gamma}$ is increasing in x, it is equivalent to show that

$$\frac{\gamma-1}{\beta-1} \ge \lambda_c = \left(\frac{\gamma}{\beta}\right)^{\frac{\sqrt{\beta\gamma}}{\sqrt{\beta\gamma}-1}} \qquad \Leftrightarrow \qquad \log(\gamma-1)-\log(\beta-1) \ge \frac{\sqrt{\beta\gamma}}{\sqrt{\beta\gamma}-1}\log\left(\frac{\gamma}{\beta}\right).$$

Let $\gamma = k^2\beta$ with $k \ge 1$. We only need to show that $r(k) \ge 0$ for $k \ge 1$, where $r(k)$ is defined as

$$r(k) := \log(\beta k^2-1) - \log(\beta-1) - \frac{2\beta k}{\beta k-1}\log k.$$

Since $r(1)=0$, it is enough to prove that $r(k)$ is increasing for $k \ge 1$. It can be easily verified as

$$\begin{aligned} r'(k) &= \frac{2\beta k}{\beta k^2-1} - \frac{2\beta}{\beta k-1} + \frac{2\beta}{(\beta k-1)^2}\log k \\ &= \frac{2\beta}{(\beta k-1)^2(\beta k^2-1)}\left((\beta k^2-1)\log k - (k-1)(\beta k-1)\right). \end{aligned}$$

So, it is sufficient to show that

$$(\beta k^2-1)\log k - (k-1)(\beta k-1) \ge 0.$$

Since $k \ge 1$, we have that $\log k \ge 1-\frac{1}{k}$. It implies that

$$(\beta k^2-1)\log k - (k-1)(\beta k-1) \ge (\beta k^2-1)(1-\frac{1}{k}) - (k-1)(\beta k-1) = \frac{(k-1)^2}{k} \ge 0.$$

This completes the proof. ◀

Then we show Lemma 15.

Proof of Lemma 15. Let $g(x) := (\beta\gamma-1)x\log\frac{\lambda_c}{x} - (\beta x+1)(x+\gamma)\log\frac{x+\gamma}{\beta x+1}$. Hence it is equivalent to show that $g(x) \le 0$ for all $0 < x < \lambda_c$. Take the derivative of $g(x)$ and we have that

$$\begin{aligned} g'(x) &= (\beta\gamma-1)(\log\frac{\lambda_c}{x}-1) - (2\beta x+\beta\gamma+1)\log\frac{x+\gamma}{\beta x+1} \\ &\quad - (\beta x+1)(x+\gamma)\left(\frac{1}{x+\gamma} - \frac{\beta}{\beta x+1}\right) \\ &= (\beta\gamma-1)\log\frac{\lambda_c}{x} - (2\beta x+\beta\gamma+1)\log\frac{x+\gamma}{\beta x+1}. \end{aligned}$$

By direct calculation, $g\left(\sqrt{\frac{\gamma}{\beta}}\right) = 0$ and $g'\left(\sqrt{\frac{\gamma}{\beta}}\right) = 0$. Then we prove (3) for the case of $0 < x < \sqrt{\frac{\gamma}{\beta}}$ and $\sqrt{\frac{\gamma}{\beta}} < x < \lambda_c$ separately.

If $0 < x < \sqrt{\frac{\gamma}{\beta}}$, it is sufficient to verify that $g'(x) > 0$. We only need to show that $g'(x)$ is decreasing since $g'\left(\sqrt{\frac{\gamma}{\beta}}\right) = 0$. It is easily verified by taking the derivative again:

$$\begin{aligned} g''(x) &= -\frac{\beta\gamma-1}{x} - 2\beta\log\frac{x+\gamma}{\beta x+1} - (2\beta x+\beta\gamma+1)\left(\frac{1}{x+\gamma} - \frac{\beta}{\beta x+1}\right) \\ &= -2\beta\log\frac{x+\gamma}{\beta x+1} - (\beta\gamma-1)\left(\frac{1}{x} - \frac{2\beta x+\beta\gamma+1}{(x+\gamma)(\beta x+1)}\right) \\ &= -2\beta\log\frac{x+\gamma}{\beta x+1} - (\beta\gamma-1)\frac{r-\beta x^2}{x(x+\gamma)(\beta x+1)} < 0, \end{aligned}$$

where the last inequality uses the fact that $\frac{x+\gamma}{\beta x+1} \geq 1$ by Lemma 14 and $x < \sqrt{\frac{\gamma}{\beta}}$.

If $\sqrt{\frac{\gamma}{\beta}} < x < \lambda_c$, then we show (3) directly. First notice that as $x \neq \sqrt{\frac{\gamma}{\beta}}$,

$$\frac{x}{(\beta x+1)(x+\gamma)} = \frac{1}{\beta x + \frac{\gamma}{x} + \beta\gamma + 1} < (\sqrt{\beta\gamma}+1)^{-2},$$

Given this, in order to get (3), it is sufficient to show that $h(x) < 0$ where

$$h(x) := \frac{\sqrt{\beta\gamma}-1}{\sqrt{\beta\gamma}+1} \log \frac{\lambda_c}{x} - \log \frac{x+\gamma}{\beta x+1}.$$

In fact, $h(x)$ is a decreasing function as

$$\begin{aligned} h'(x) &= -\frac{\sqrt{\beta\gamma}-1}{x(\sqrt{\beta\gamma}+1)} - \frac{1}{x+\gamma} + \frac{\beta}{\beta x+1} \\ &= -\frac{(\sqrt{\beta\gamma}-1)\left((x+\gamma)(\beta x+1) - (\sqrt{\beta\gamma}+1)^2 x\right)}{x(\sqrt{\beta\gamma}+1)(x+\gamma)(\beta x+1)} \\ &= -\frac{(\sqrt{\beta\gamma}-1)\left(\sqrt{\beta}x - \sqrt{\gamma}\right)^2}{x(\sqrt{\beta\gamma}+1)(x+\gamma)(\beta x+1)} \leq 0. \end{aligned}$$

Notice that $h\left(\sqrt{\frac{\gamma}{\beta}}\right) = 0$. It implies that $h(x) < 0$ for all $x > \sqrt{\frac{\gamma}{\beta}}$. This completes the proof. ◀

Acknowledgement. We thank Liang Li, Jingcheng Liu, and Chihao Zhang for some stimulating discussion. In particular, the example of the 5-7 tree in Section 5 is an outcome from such discussion. We also thank organizers of the "IMA-GaTech Workshop on the Power of Randomness in Computation" in March 2015. The current work stems from discussions during the workshop.

References

1 Jin-Yi Cai, Andreas Galanis, Leslie Ann Goldberg, Heng Guo, Mark Jerrum, Daniel Štefankovič, and Eric Vigoda. #BIS-hardness for 2-spin systems on bipartite bounded degree graphs in the tree non-uniqueness region. In *RANDOM*, pages 582–595, 2014.

2 Jin-Yi Cai and Michael Kowalczyk. Spin systems on k-regular graphs with complex edge functions. *Theor. Comput. Sci.*, 461:2–16, 2012.

3 Martin E. Dyer, Leslie Ann Goldberg, Catherine S. Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2003.

4 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic Ising and Hard-Core models. *CoRR*,, 2012. URL: `http://arxiv.org/abs/1203.2226`.

5 Hans-Otto Georgii. *Gibbs Measures and Phase Transitions*, volume 9 of *De Gruyter Studies in Mathematics*. de Gruyter, Berlin, second edition, 2011.

6 Leslie Ann Goldberg and Mark Jerrum. The complexity of ferromagnetic Ising with local fields. *Combinatorics, Probability & Computing*, 16(1):43–61, 2007.

7 Leslie Ann Goldberg, Mark Jerrum, and Mike Paterson. The computational complexity of two-state spin systems. *Random Struct. Algorithms*, 23(2):133–154, 2003.

8 Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM J. Comput.*, 22(5):1087–1116, 1993.

9 Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

10 Frank P. Kelly. Stochastic models of computer communication systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(3):379–395, 1985.

11 Liang Li, Pinyan Lu, and Yitong Yin. Approximate counting via correlation decay in spin systems. In *SODA*, pages 922–940, 2012.

12 Liang Li, Pinyan Lu, and Yitong Yin. Correlation decay up to uniqueness in spin systems. In *SODA*, pages 67–84, 2013.

13 Jingcheng Liu, Pinyan Lu, and Chihao Zhang. The complexity of ferromagnetic two-spin systems with external fields. In *RANDOM*, pages 843–856, 2014.

14 Russell Lyons. The Ising model and percolation on trees and tree-like graphs. *Comm. Math. Phys.*, 125(2):337–353, 1989.

15 Elchanan Mossel and Allan Sly. Exact thresholds for Ising-Gibbs samplers on general graphs. *Annals of Probability*, 41(1):294–328, 2013.

16 Alistair Sinclair, Piyush Srivastava, Daniel Štefankovič, and Yitong Yin. Spatial mixing and the connective constant: Optimal bounds. In *SODA*, pages 1549–1563, 2015.

17 Alistair Sinclair, Piyush Srivastava, and Marc Thurley. Approximation algorithms for two-state anti-ferromagnetic spin systems on bounded degree graphs. In *SODA*, pages 941–953, 2012.

18 Allan Sly and Nike Sun. The computational hardness of counting in two-spin models on d-regular graphs. *The Annals of Probability*, 42(6):2383–2416, 2014.

19 Dror Weitz. Counting independent sets up to the tree threshold. In *STOC*, pages 140–149, 2006.

20 Jinshan Zhang, Heng Liang, and Fengshan Bai. Approximating partition functions of the two-state spin system. *Inf. Process. Lett.*, 111(14):702–710, 2011.

On Polynomial Approximations to AC^0

Prahladh Harsha[1] **and Srikanth Srinivasan**[2]

1 **TIFR, Mumbai, India**
`prahladh@tifr.res.in`

2 **Dept. of Mathematics, IIT Bombay, Mumbai, India**
`srikanth@math.iitb.ac.in`

Abstract

We make progress on some questions related to polynomial approximations of AC^0. It is known, from the works of Tarui (*Theoret. Comput. Sci.* 1993) and Beigel, Reingold, and Spielman (*Proc. 6th CCC* 1991), that any AC^0 circuit of size s and depth d has an ε-error probabilistic polynomial over the reals of degree $(\log(s/\varepsilon))^{O(d)}$. We improve this upper bound to $(\log s)^{O(d)} \cdot \log(1/\varepsilon)$, which is much better for small values of ε.

We give an application of this result by using it to resolve a question posed by Tal (ECCC 2014): we show that $(\log s)^{O(d)} \cdot \log(1/\varepsilon)$-wise independence fools AC^0, improving on Tal's strengthening of Braverman's theorem (*J. ACM* 2010) that $(\log(s/\varepsilon))^{O(d)}$-wise independence fools AC^0. Up to the constant implicit in the $O(d)$, our result is tight. As far as we know, this is the first PRG construction for AC^0 that achieves optimal dependence on the error ε.

We also prove lower bounds on the best polynomial approximations to AC^0. We show that any polynomial approximating the OR function on n bits to a small constant error must have degree at least $\widetilde{\Omega}(\sqrt{\log n})$. This result improves exponentially on a recent lower bound demonstrated by Meka, Nguyen, and Vu (arXiv 2015).

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation

Keywords and phrases Constant-depth Boolean circuits, Polynomials over reals, pseudo-random generators, k-wise independence

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.32

1 Motivation and Results

We use $\mathrm{AC}^0(s,d)$ to denote AC^0 circuits of size s and depth d.

Polynomial approximations to AC^0

In his breakthrough work on proving lower bounds for the class $\mathrm{AC}^0[\oplus]$, Razborov [17] studied how well small circuits can be approximated by low-degree polynomials. We recall (an equivalent version of) his notion of polynomial approximation over the reals.

An *ε-error probabilistic polynomial* (over the reals) for a circuit $C(x_1,\ldots,x_n)$ is a random polynomial $\mathbf{P}(x_1,\ldots,x_n) \in \mathbb{R}[x_1,\ldots,x_n]$ such that for any $a \in \{0,1\}^n$, we have $\Pr_{\mathbf{P}}[C(a) \neq \mathbf{P}(a)] \leq \varepsilon$. Further, we say that $\mathbf{P}$ has degree D and $\|\mathbf{P}\|_\infty \leq L$ if $\mathbf{P}$ is supported on polynomials P of degree at most D and L_∞ norm at most L (i.e. polynomials P such that $\max_{a\in\{0,1\}^n} |P(a)| \leq L$). If there is such a $\mathbf{P}$ for C, we say that C has an ε-error probabilistic degree at most D.

It is well-known [22, 21, 2] that any circuit $C \in \mathrm{AC}^0(s,d)$ has an ε-error probabilistic polynomial $\mathbf{P}$ of degree $(\log(s/\varepsilon))^{O(d)}$ and satisfying $\|\mathbf{P}\|_\infty < \exp((\log s/\varepsilon)^{O(d)})$. This can be used to prove, for example [19], (a slightly weaker version of) Håstad's theorem [6] that says that Parity does not have subexponential-sized AC^0 circuits. It also plays an important role in Braverman's theorem [3] that shows that polylog-wise independence fools AC^0 circuits.

Upper bounds for probabilistic polynomials

We show a general result regarding error reduction of probabilistic polynomials over the reals.

▶ **Theorem 1.** *Suppose $f : \{0,1\}^n \to \{0,1\}$ has a $(\frac{1}{2} - \delta)$-error probabilistic polynomial $\mathbf{P}$ of degree D and L_∞ norm at most $L \geq 2$. Then, for any $\varepsilon > 0$, f has an ε-error probabilistic polynomial of degree at most $O\left(\frac{D}{\delta^2}\log(1/\varepsilon)\right)$ and L_∞ norm at most $L^{O\left(\frac{1}{\delta^2}\log\frac{1}{\varepsilon}\right)}$.*

Applying the above result to (1/10)-error probabilistic polynomials for AC^0 gives us small-error probabilistic polynomials for AC^0 with better parameters.

▶ **Theorem 2.** *Let C be any AC^0 circuit of size s and depth d. Let $\varepsilon > 0$ be any parameter. The circuit C has an ε-error probabilistic polynomial $\mathbf{P}$ of degree $(\log s)^{O(d)} \cdot \log(1/\varepsilon)$ and $\|\mathbf{P}\|_\infty \leq \exp((\log s)^{O(d)}\log(1/\varepsilon))$.*

Similar results on probabilistic polynomials were obtained over $\mathbb{F}_2$ (for the larger class of $\mathrm{AC}^0[\oplus]$ circuits) by Kopparty and Srinivasan [9] and extended to all fixed non-zero characteristics by Oliveira and Santhanam [16]. They have also found applications in the works of Williams [24] – for the purposes of obtaining better algorithms for satisfiability problems – and Oliveira and Santhanam [16], for proving lower bounds on compression by bounded-depth circuits. However, as far as we know, no corresponding results were observed over the reals until now.

The above theorem was motivated by an application to constructing pseudorandom generators (PRGs) for AC^0. As mentioned above, it was shown by Braverman [3] that AC^0 is fooled by polylog-wise independence.The proof of Braverman's theorem proceeds by constructing certain approximating polynomials for AC^0, which in turn depends on two previous polynomial approximation results for this circuit class. The first of these is the L_2-approximation result of Linial, Mansour and Nisan [10] which is based on the classical Håstad Switching Lemma [6], and the second is the above mentioned result of Tarui [21] and Beigel et al. [2]. Using these constructions, Braverman showed that $\mathrm{AC}^0(s,d)$ is ε-fooled by $(\log(s/\varepsilon))^{O(d^2)}$-wise independence.

An example due to Mansour appearing in the work of Luby and Veličković [12] demonstrated that $(\log s)^{d-1}\log(1/\varepsilon)$-wise independence is *necessary* to ε-fool $\mathrm{AC}^0(s,d)$. This leads naturally to the question of showing tight bounds for the amount of independence required to fool $\mathrm{AC}^0(s,d)$.

Using an improved switching lemma due to Håstad [7] (see also the work of Impagliazzo, Matthews, and Paturi [8]), Tal [20] gave an improved version of the L_2-approximation result of Linial et al. [10], and used this to improve the parameters of Braverman's theorem. Specifically, he showed that $(\log(s/\varepsilon))^{O(d)}$-wise independence fools AC^0.

Tal asked if the dependence on ε in this result could be made to match the limit given by Mansour's example. Formally, he asked if $(\log s)^{O(d)} \cdot \log(1/\varepsilon)$-wise independence fools $\mathrm{AC}^0(s,d)$. In this work, we are able to answer this question in the affirmative (Corollary 12 below). Up to the constant implicit in the $O(d)$, our result is optimal for all $\varepsilon > 0$.

Comparison to other PRGs for AC^0

Using standard constructions of k-wise independent probability distributions, the above result gives explicit PRGs with seedlength $(\log s)^{O(d)} \cdot \log(1/\varepsilon)$ for fooling circuits from $\mathrm{AC}^0(s,d)$.

It is easy to see that this seedlength cannot be improved beyond $\Omega(\log(1/\varepsilon))$ and hence that our result is optimal in terms of the error parameter ε.

It is also instructive to see how well this compares to general (i.e. not based on limited independence) PRG constructions for AC^0. Using the standard Hardness-to-Randomness paradigm of Nisan and Wigderson [14] and the best known average case lower bounds for AC^0 [8, 7], it is easy to obtain PRGs of seedlength $(\log s)^{O(d)} \cdot (\log(1/\varepsilon))^2$ for $AC^0(s,d)$. Furthermore, the Nisan-Wigderson paradigm cannot yield PRGs of seedlength less than $(\log(1/\varepsilon))^2$ given our current state of knowledge regarding circuit lower bounds (see Appendix A for details). Another recent PRG construction for $AC^0(s,d)$ due to Trevisan and Xue [23] has seedlength $(\log(s/\varepsilon))^{d+O(1)}$.

The reader will note that both constructions are suboptimal in terms of the dependence on ε (though both are better than ours in terms of dependence on s and d). Interestingly, as far as we know, our construction is the first that achieves an optimal dependence on ε.

Lower bounds for probabilistic polynomials

We can also ask if our result can be strengthened to yield a seedlength of $(\log s)^{d+O(1)} \cdot \log(1/\varepsilon)$, which would generalize both our current construction and that of Trevisan and Xue [23], and almost match Mansour's lower bound as well. Such a strengthening could conceivably be obtained by improving the polynomial approximation results for AC^0 [21, 2]. Razborov [17] observed that to obtain good approximations for $AC^0(s,d)$, it suffices to approximate the OR function on s bits efficiently. Therefore, we study the probabilistic degree of the OR function.

Beigel, Reingold and Spielman [2] and Tarui [21] showed that the OR function on n bits can be ε-approximated by a polynomial of degree $O((\log n) \cdot \log(1/\varepsilon))$. While it is easy to show that the dependence on ε in this result is tight (in fact for *any field*), for a long time, it was not known if *any* dependence on n is necessary over the reals[1]. Recently, Meka, Nguyen and Vu [13] showed that any *constant error* probabilistic polynomial for the OR function over the reals must have degree $\widetilde{\Omega}(\log \log n)$ and hence the dependence on the parameter n is unavoidable. We further improve the bound of Meka et al. exponentially to $\widetilde{\Omega}(\sqrt{\log n})$, which is only a quadratic factor away from the upper bound.

1.1 Proof ideas

Here, we describe the ideas behind the proofs of the main results.

The proof of Theorem 1 is extremely simple. A natural strategy to reduce the error of a (constant-error, say) probabilistic polynomial $\mathbf{P}$ is to sample it independently $\ell = O(\log(1/\varepsilon))$ times to obtain polynomials $\mathbf{P}_1, \ldots, \mathbf{P}_\ell$ and then take the Majority vote among the $\mathbf{P}_i$s, which can be simulated by composing with a multilinear polynomial of degree ℓ. Indeed, this is exactly what Kopparty and Srinivasan [9] do in an earlier work to obtain ε-error probabilistic polynomials over $\mathbb{F}_2$.

Over the reals, it is not completely clear that this strategy works, since the polynomials $\mathbf{P}_i$ need not output a Boolean value when they err and hence it is not clear what taking a "Majority vote" means. Nevertheless, we observe that composing with the multilinear Majority polynomial continues to work since this polynomial has the nice property that setting more than half of its input bits to a constant $b \in \{0,1\}$ causes the polynomial to

[1] In fact, for finite fields of constant size, Razborov [17] showed that the ε-error probabilistic degree of OR is $O(\log(1/\varepsilon))$, independent of the number of input bits.

collapse to the constant polynomial b, which is oblivious to the values of the unset inputs (that could even be non-Boolean and possibly arbitrarily large real numbers).

As mentioned already above, Theorem 1, along with standard constructions of probablistic polynomials for $\mathrm{AC}^0(s, d)$ in the constant-error regime, directly proves Theorem 2. We can more or less plug this result into Tal's proof [20] of Braverman's theorem to obtain better parameters for the amount of independence required to fool AC^0. The only additional idea required is to ensure that the inputs where the probabilistic polynomial computes the correct value are certified by a small AC^0 circuit. While a small AC^0 circuit *cannot* compute the Majority vote above, it turns out that an "Approximate Majority" [1] is sufficient for this purpose, and this can be done in AC^0.

We now describe the proof of the degree lower bound for ε-error probabilistic polynomials computing the OR function on n variables to a small constant-error (say 1/10). It is known that this can be done over fields of *constant* characteristic with constant degree [17] and over the reals with degree $O(\log n)$ [21, 2]. Hence any technique for proving lower bounds growing with n will have to use a technique specific to large characteristic.

The work of Razborov and Viola [18] introduced such a technique to the theoretical computer science literature to show that no low degree polynomial over the reals can compute the Parity function on more than half its inputs. The main technique was an anti-concentration lemma generalizing classical theorems of Littlewood-Offord and Erdős [11, 5] that state that any linear function of at least r Boolean variables takes any fixed value on a uniformly random input with probability at most $O(1/\sqrt{r})$. In particular, it cannot approximate a Boolean function well unless r is very small. Razborov and Viola, building on the work of Costello, Tao, and Vu [4], proved a generalization of this statement to low-degree multivariate polynomials that contain at least r disjoint monomials of maximum degree.

More recently, Meka, Nguyen, and Vu [13] proved an improved (and near-optimal) version of the anti-concentration lemma of Razborov and Viola and used this to show better lower bounds for the Parity function. Additionally, they were also able to show that any constant-error probabilistic polynomial for the OR function must have degree $\widetilde{\Omega}(\log\log n)$. We use their anti-concentration lemma with a more efficient restriction argument to prove a lower bound of $\widetilde{\Omega}(\sqrt{\log n})$. We describe the outline of this restriction argument next.

To prove a lower bound of D on the probabilistic degree of some function it suffices (and is also necessary, by standard duality arguments) to obtain a distribution under which the function is hard to approximate by any polynomial of degree less than D. While some functions have 'obvious' hard distributions (such as the Parity function, which is random self-reducible w.r.t. the uniform distribution), the OR function is not one such, since it takes value 0 only on one input. Some obvious candidates (such as the uniform distribution or a convex combination of the uniform distribution along with the distribution that puts all its mass on the all 0s input) can actually be shown to be easy for the OR function. The hard distribution we use is motivated by the polynomial constructions of [2, 21] and is as follows: with probability 1/2 choose the all 0s input and with probability 1/2 choose a uniformly random $i \in [\log n]$ and then choose a random input of weight[2] $n/2^i$. The hard distribution chosen by Meka et al. is similar, but sparser than the distribution we use (it is only concentrated on $\log\log n$ levels of the hypercube whereas our distribution is concentrated on $\log n$ levels).

We now argue that any polynomial q approximating the OR function w.r.t. this distribution must be of large degree as follows. First of all, since there is a considerable amount of

[2] We will actually use the product distribution where each bit is set to 1 with probability $\frac{1}{2^i}$, which puts most of its mass on inputs of weight *close* to $n/2^i$, but we blur this distinction here.

mass on the all 0s input, we can assume that q takes value 0 on this input. Now, we consider the distribution that is uniformly distributed on inputs of Hamming weight $n/2$. We know that the OR function is always 1 on these inputs, which means that q is *not* anti-concentrated on inputs from this distribution. Hence, by the anti-concentration lemma due to Meka et al., any maximal disjoint set of maximum degree monomials in q cannot have too many monomials, say more than r. In particular, setting all the variables V in such a set of monomials – there are at most rD variables in V – to 0 reduces the degree of the polynomial by 1. The important observation is that this naturally happens with high probability when we use the distribution that is uniformly distributed on inputs of weight $\approx n/rD$, since each variable is set to 1 only with probability $\approx 1/rD$. Further, we can simulate the uniform distribution on (say) inputs of weight n/rD by first sampling a set S of size $2n/rD$ and setting the bits outside S to 0 – this sets all the variables in V with good probability and thus reduces the degree of q – and then choosing a random set of $|S|/2$ inputs to set to 1. We are now exactly in the situation we were at the beginning of this paragraph, except for the fact that the degree of q is smaller.

Continuing in this way, we eventually obtain a constant polynomial q that computes the OR function on some non-zero inputs from the hypercube, which means that it must be the constant polynomial 1. However, this contradicts the fact that q takes value 0 on the all 0s input and this proves the theorem.

2 Improved probabilistic polynomials and PRGs for AC^0

2.1 The construction of probabilistic polynomials

Notation

Let $P \in \mathbb{R}[x_1, \ldots, x_\ell]$. Given a set $S \subseteq [\ell]$ and a partial assignment $\sigma : S \to \{0,1\}$, we define $P|_\sigma$ to be the polynomial obtained by setting all the bits in S according to σ. In the case that σ sets all the variables in S to a constant $b \in \{0,1\}$, we use $P|_{S\mapsto b}$ instead of $P|_\sigma$. For a function $f : \{0,1\}^\ell \to \{0,1\}$, we define $f|_\sigma$ and $f|_{S\mapsto b}$ similarly.

We define the *weight* of P, denoted $w(P)$, to be the sum of the absolute values of all the coefficients of P.

▶ **Definition 3.** Let $P \in \mathbb{R}[x_1, \ldots, x_\ell]$ and say r is a parameter from $[\ell]$. We say that P is an *ℓ-pseudo-majority* if for r being the least integer greater than $\ell/2$ and any $S \in \binom{[\ell]}{r}$ and $b \in \{0,1\}$, the polynomial $P|_{S\mapsto b}$ is the constant polynomial b.

We show below that the multilinear polynomial representing the Majority function is an ℓ-pseudo-majority of weight $2^{O(\ell)}$.

Before we prove that this construction works, we need a few standard facts about polynomials.

▶ **Fact 4.** *Any Boolean function $f : \{0,1\}^\ell \to \{0,1\}$ can be represented uniquely by a multilinear polynomial $P[x_1, \ldots, x_\ell]$ in the sense that for all $a \in \{0,1\}^n$, we have $P(a) = f(a)$. Furthermore, $w(P) = 2^{O(\ell)}$.*

The uniqueness in the fact above yields the following observation.

▶ **Lemma 5.** *Let $f : \{0,1\}^\ell \to \{0,1\}$ and P be the corresponding unique multilinear polynomial guaranteed by Fact 4. If $\sigma : S \to \{0,1\}$ is a partial assignment such that $f|_\sigma$ is the constant function $b \in \{0,1\}$, then $P|_\sigma$ is formally the constant polynomial b.*

Proof. Follows from the fact that $P|_\sigma$ is a multilinear polynomial representing the constant function b on the variables not in S and the uniqueness part of Fact 4. ◀

▶ Remark. Note that the hypothesis of the lemma above is that $f|_\sigma(a) = b$ for all Boolean assignments a to the remaining variables. However, the conclusion yields a stronger conclusion for the polynomial P: namely, we show that $P|_\sigma$ takes value b on *any* assignment $a \in \mathbb{R}^{\ell-|S|}$ to the remaining variables, and not just Boolean assignments. It is this fact that we will use in applications below.

For $\ell \in \mathbb{N}$, define the Boolean function M_ℓ to be the Majority function: i.e., $M_\ell(x) = 1$ iff the Hamming weight of x is strictly greater than $\ell/2$. Note that for any $S \subseteq \{x_1, \ldots, x_\ell\}$ of size greater than $\ell/2$ and any $b \in \{0,1\}$, $M_\ell|_{S \mapsto b}$ is the constant function b.

Let P_ℓ be the multilinear polynomial representing M_ℓ guaranteed by Fact 4. Applying Lemma 5 to the pair M_ℓ and P_ℓ, we obtain the following corollary.

▶ **Corollary 6.** *For any $\ell \in \mathbb{N}$, there exist ℓ-pseudo-majorities of degree ℓ and weight $2^{O(\ell)}$.*

We now prove Theorem 1. We will follow the proof of [9, Lemma 10], but some additional justification will be required since we are working over the reals and not over $\mathbb{F}_2$ as in [9].

Proof of Theorem 1. We set $\ell = \frac{A}{\delta^2}\log(\frac{1}{\varepsilon})$ for a constant $A > 0$ to be fixed later. Let $\mathbf{P}_1, \ldots, \mathbf{P}_\ell$ be ℓ mutually independent copies of the probabilistic polynomial $\mathbf{P}$. Let $r = \lfloor \frac{\ell}{2} \rfloor$. Fix an ℓ-pseudo-majority Q as guaranteed by Corollary 6. The final probabilistic polynomial is $\mathbf{R} = Q(\mathbf{P}_1, \ldots, \mathbf{P}_\ell)$.

The degree of $\mathbf{R}$ is at most $\deg(Q) \cdot \deg(\mathbf{P}) \leq O\left(\frac{D}{\delta^2}\log(\frac{1}{\varepsilon})\right)$. Moreover, it can be seen that the $\|\mathbf{R}\|_\infty \leq w(Q) \cdot L^{\deg(Q)} \leq (2L)^{O(\ell)} \leq L^{O(\ell)}$ since $L \geq 2$.

Finally, we see that for any $a \in \{0,1\}^n$, $\mathbf{R}(a) = f(a)$ unless at least for r many $i \in [\ell]$, we have $\mathbf{P}_i(a) \neq f(a)$. By a Chernoff bound, the probability of this is at most ε as long as A is chosen to be a suitably large constant. Hence, $\mathbf{R}$ is indeed an ε-error probabilistic polynomial for f. ◀

Theorem 2 immediately follows from the above and standard probabilistic polynomials for AC^0 from [22, 21, 2]. However, for our applications to PRGs for AC^0, we need a slightly stronger statement, which we prove below.

▶ **Definition 7** (Probabilistic polynomial with witness). An ε-error probabilistic polynomial for circuit $C(x_1, \ldots, x_n)$ with witness (ε-error PPW for short) is a pair $(\mathbf{P}, \boldsymbol{\mathcal{E}})$ of random variables such that $\mathbf{P}$ is a randomized polynomial and $\boldsymbol{\mathcal{E}}$ is a randomized circuit (both on n Boolean variables) such that for any input $a \in \{0,1\}^n$, we have

- $\Pr_{\boldsymbol{\mathcal{E}}}[\boldsymbol{\mathcal{E}}(a) = 1] \leq \varepsilon$,
- For any fixing $(P, \mathcal{E})$ of $(\mathbf{P}, \boldsymbol{\mathcal{E}})$, we have $\mathcal{E}(a) = 0 \Rightarrow P(a) = C(a)$.

In particular, this implies that $\mathbf{P}$ is an ε-error probabilistic polynomial for C.

We say that $\boldsymbol{\mathcal{E}}$ belongs to a circuit class $\mathcal{C}$ if it is supported on circuits from class $\mathcal{C}$.

The above notion was introduced in Braverman [3] who proved the following lemma, building on earlier works of [22, 21, 2].

▶ **Lemma 8** ([3, Lemma 8, Proposition 9]). *Fix parameters $s, d \in \mathbb{N}$ and $\varepsilon > 0$. Any AC^0 circuit C of size s and depth d has an ε-error PPW $(\mathbf{P}, \boldsymbol{\mathcal{E}})$ where*

- $\deg(\mathbf{P}) \leq (\log(s/\varepsilon))^{O(d)}$ *and* $\|\mathbf{P}\|_\infty \leq \exp\left((\log(s/\varepsilon))^{O(d)}\right)$,
- $\boldsymbol{\mathcal{E}} \in \mathrm{AC}^0(\mathrm{poly}(s\log(1/\varepsilon)), d+3)$.

We show the following variant of the above lemma, which is an improvement in terms of degree and the L_∞ norm of the probabilistic polynomial for small ε.

▶ **Lemma 9.** *Fix parameters $s, d \in \mathbb{N}$ and $\varepsilon > 0$. Any AC^0 circuit C of size s and depth d has an ε-error PPW $(\mathbf{P}, \boldsymbol{\mathcal{E}})$ where*
- $\deg(\mathbf{P}) \leq (\log s)^{O(d)} \cdot \log(1/\varepsilon)$ *and* $\|\mathbf{P}\|_\infty \leq \exp\left((\log s)^{O(d)} \log(1/\varepsilon)\right)$,
- $\boldsymbol{\mathcal{E}} \in \mathrm{AC}^0(\mathrm{poly}(s \log(1/\varepsilon)), d + O(1))$.

Before we begin the proof, we state one more lemma from the literature. Given an integer parameter ℓ and real parameters $\alpha, \beta \in [0,1]$ with $\alpha < \beta$, we will call a function $f : \{0,1\}^\ell \to \{0,1\}$ an (ℓ, α, β)-*approximate majority* if $f(x) = 0$ for any input of Hamming weight at most $\alpha\ell$ and $f(x) = 1$ for any input of Hamming weight at least $\beta\ell$. The following is a result of Ajtai and Ben-Or [1].

▶ **Lemma 10** (Ajtai and Ben-Or [1]). *Fix any constants $\alpha < \beta$. Then, for all $\ell \in \mathbb{N}$, there is an (ℓ, α, β)-approximate majority which has an AC^0 circuit of size $\mathrm{poly}(\ell)$ and depth 3.*

We now prove Lemma 9. The proof is similar to that of Theorem 1 above, but we also need to obtain a witness circuit for our probabilistic polynomial.

Proof of Lemma 9. Let $\ell = A \log(1/\varepsilon)$ for a large constant A to be chosen later. W.l.o.g. assume that ℓ is even. Let $r = \lceil \ell/2 \rceil + 1$ and let $Q(x_1, \ldots, x_\ell)$ be the ℓ-pseudo-majority guaranteed by Corollary 6. Let $k = \ell/4$. By Lemma 10, there is an AC^0 circuit C_1 of size $\mathrm{poly}(\ell)$ and depth 3 that computes an $(\ell, 1/4, 2/5)$-approximate majority.

Let $(\mathbf{P}_1, \boldsymbol{\mathcal{E}}_1), \ldots, (\mathbf{P}_\ell, \boldsymbol{\mathcal{E}}_\ell)$ be *independent* copies of the $(1/8)$-error PPW guaranteed by Lemma 8. The final PPW is $(\mathbf{P}, \boldsymbol{\mathcal{E}})$ where $\mathbf{P} = Q(\mathbf{P}_1, \ldots, \mathbf{P}_\ell)$ and $\boldsymbol{\mathcal{E}} = C_1(\boldsymbol{\mathcal{E}}_1, \ldots, \boldsymbol{\mathcal{E}}_\ell)$. We show that this PPW has the required properties.

First of all, we know that on any input a to the circuit C and for any $i \in [\ell]$, the probability that $\boldsymbol{\mathcal{E}}_i(a) = 1$ is at most $1/8$. Thus, the expected number of $\boldsymbol{\mathcal{E}}_i$ that output 1 is at most $\ell/8$. However, for $\boldsymbol{\mathcal{E}}(a)$ to be 1, at least $\ell/4$ many $\boldsymbol{\mathcal{E}}_i(a)$ should be 1. By a Chernoff bound, the probability of this event is at most $\exp(-\Omega(\ell)) < \varepsilon$ for a large enough constant A.

Now, we need to argue that if $\boldsymbol{\mathcal{E}}(a) = 0$, then $\mathbf{P}(a) = Q(\mathbf{P}_1(a), \ldots, \mathbf{P}_\ell(a)) = C(a)$. Say $C(a) = b \in \{0,1\}$. If $\boldsymbol{\mathcal{E}}(a) = 0$, then we know that the number of $\boldsymbol{\mathcal{E}}_i(a)$ that are 0 is at least $3\ell/5$; let I denote the set of these i. By the definition of PPWs, we know that for each $i \in I$, we have $\mathbf{P}_i(a) = b$ and hence at least $3\ell/5 > r$ many inputs of Q are set to b. Since Q is an ℓ-pseudo-majority, we must have $Q(\mathbf{P}_1(a), \ldots, \mathbf{P}_\ell(a)) = b$. This concludes the proof that $(\mathbf{P}, \boldsymbol{\mathcal{E}})$ is indeed an ε-error PPW for C.

Note that $\deg(\mathbf{P}) \leq \deg(Q) \cdot \max_i \deg(\mathbf{P}_i) \leq (\log s)^{O(d)} \log(1/\varepsilon)$. Also, it can be seen that

$$\|\mathbf{P}\|_\infty \leq w(Q) \cdot (\max_{i \in [\ell]} \|\mathbf{P}_i\|_\infty)^{\deg(Q)} \leq \exp\left((\log s)^{O(d)} \log(1/\varepsilon)\right).$$

Thus, $\mathbf{P}$ has the required properties. The size and depth properties of $\boldsymbol{\mathcal{E}}$ follow trivially from its definition. This concludes the proof of the lemma. ◀

2.2 Application to PRGs for AC^0

The connection between probabilistic polynomials and PRGs for AC^0 is encapsulated in the following theorem (which is an easy observation from the works of Braverman and Tal):

▶ **Theorem 11** (Braverman [3],Tal [20]). *Let $s, d \in \mathbb{N}$ and $\varepsilon > 0$. Suppose that any AC^0 circuit of size s and depth d has an $(\varepsilon/2)$-error PPW $(\mathbf{P}, \boldsymbol{\mathcal{E}})$ such that*

- $\deg(\mathbf{P}) = D$, $\|\mathbf{P}\|_\infty \leq L$,
- $\boldsymbol{\mathcal{E}} \in \mathrm{AC}^0(s_1, d_1)$,

Then, AC^0 circuits of size s and depth d can be ε-fooled by $k(s, d, \varepsilon)$-wise independence, where

$$k(s, d, \varepsilon) = O(D) + (\log s_1)^{O(d_1)} \cdot (\log(1/\varepsilon) + \log L)$$

Note that the theorem above is trivial when $\log(1/\varepsilon) > s$ since any AC^0 circuit of size s is trivially fooled by an s-wise independent distribution. Hence, the theorem is non-trivial only when $\log(1/\varepsilon) \leq s$. In this case, using Lemma 9 and the theorem above, we immediately get

▶ **Corollary 12.** *Fix parameters $s, d \in \mathbb{N}$ and $\varepsilon > 0$. Any circuit $C \in \mathrm{AC}^0(s, d)$ can be ε-fooled by any distribution that is $(\log s)^{O(d)} \log(1/\varepsilon)$-wise independent.*

▶ Remark. A close look at the above proof (including the details of Lemma 8 and Theorem 11) shows that the amount of independence required to ε-fool $\mathrm{AC}^0(s, d)$ is $(\log s)^{3d+O(1)} \cdot \log(1/\varepsilon)$. Avishay Tal (personal communication) showed that the above can be further improved to $(\log s)^{2.5d+O(1)} \cdot \log(1/\varepsilon)$-wise independence. It is open if this can be further strengthened to, say, $(\log s)^{d+O(1)} \cdot \log(1/\varepsilon)$ or even $(\log s)^{d-1} \cdot \log(1/\varepsilon)$, matching the lower bound due to Mansour [12].

3 The probabilistic degree of OR

Notation

For $i \geq 1$ and a set of Boolean variables X, let μ_i^X be the product distribution on $\{0, 1\}^X$ defined so that for each $x \in X$, the probability that $x = 1$ is 2^{-i}. We also use $\mathcal{U}_X$ to denote μ_1^X, the uniform distribution over $\{0, 1\}^X$. The OR function on the variables in X is denoted OR_X.

We want to show:

▶ **Theorem 13.** *Let $|X_0| = n$. The $1/8$-error probabilistic degree of OR_{X_0} is $\Omega\left(\frac{\sqrt{\log n}}{(\log \log n)^{3/2}}\right)$.*

▶ Remark. Though the theorem is stated for error $1/8$, it is not hard to see that it holds (with constant factor losses) as long as the error is bounded by $1/2 - \Omega(1)$. One way to see this is to appeal to Theorem 2. Another way is to do a simpler error reduction specific to the OR function as we do in the proof of Theorem 13.

In order to prove Theorem 13, we use an anti-concentration lemma due to Meka, Nguyen and Vu [13][3] coupled with a random restriction argument inspired by the work of Razborov and Viola [18].

▶ **Lemma 14** (Meka, Nguyen, and Vu [13]). *There exists an absolute constant $B > 0$ so that the following holds. Let $p(x) \in \mathbb{R}[X]$ be a degree d multilinear polynomial with at least r disjoint degree d terms. Then $\Pr_{x \sim \mathcal{U}_X}[p(x) = 0] \leq Bd^{4/3} r^{-\frac{1}{4d+1}} \sqrt{\log r}$.*

[3] The result of Meka et al. is actually stated for polynomials over the *Fourier basis* of Parity functions (see, e.g., the book of O'Donnell [15]). However, it is an easy observation that a polynomial of degree d has r disjoint terms of degree d in the standard monomial basis if and only if it has r disjoint terms of degree d in the Fourier basis. Hence, the result holds in the standard basis as well.

Given a polynomial $q \in \mathbb{R}[X]$, we denote by $\mathrm{Err}_i^X(q)$ the error of polynomial q w.r.t. distribution μ_i^X. Formally,

$$\mathrm{Err}_i^X(q) = \Pr_{x \sim \mu_i^X}[q(x) \neq \mathrm{OR}_X(x)]$$

For a set of variables X, $\ell \in \mathbb{N}$ and $\delta \in \mathbb{R}^{\geq 0}$, call a polynomial $q \in \mathbb{R}[X]$ (X, ℓ, δ)-good if

$$\mathop{\mathbf{E}}_{i \in [\ell]}[\mathrm{Err}_i^X(q)] \leq \delta.$$

A random restriction on the variable set X with $*$-probability $p \in [0, 1]$ will be a function $\rho : X \to \{*, 0\}$ with each variable set independently to $*$ with probability p and to 0 otherwise. We use X_ρ to denote $\rho^{-1}(*)$. The restriction of a polynomial q under ρ is denoted $q|_\rho$.

▶ **Observation 15.** *Let $q \in \mathbb{R}[X]$ and ρ be a random restriction on the variable set X with $*$-probability $p = \frac{1}{2^b}$ where $b \in \mathbb{N}$. For any $i \geq 1$,*

$$\mathop{\boldsymbol{E}}_{\rho}[\mathrm{Err}_i^{X_\rho}(q|_\rho)] = \mathrm{Err}_{i+b}^X(q)$$

(I.e., setting bits independently to 1 with probability $\frac{1}{2^{i+b}}$ is the same as first applying a random restriction with $*$-probability $\frac{1}{2^b}$ and then setting each surviving variable to 1 with probability $\frac{1}{2^i}$.)

3.1 Proof of Theorem 13

We argue by contradiction. Let $\mathbf{P}$ be a 1/8-error probabilistic polynomial for OR_{X_0} of degree $D < \sqrt{\log n}/A(\log\log n)^{3/2}$ for some absolute constant $A > 0$ that we will fix in Claim 16. In particular, we have

$$\Pr_{\mathbf{P}}[\mathbf{P}(0, 0, \ldots, 0) \neq 0] \leq \frac{1}{8}$$

We discard all polynomials q such that $q(0, 0, \ldots, 0) \neq 0$ from the distribution underlying $\mathbf{P}$ (e.g. if such a bad polynomial is sampled, then we could just output 0). The resulting probabilistic polynomial $\mathbf{P}'$ is supported only on polynomials $q \in \mathbb{R}[X_0]$ such that $q(0, 0, \ldots, 0) = 0$ and further, $\mathbf{P}'$ is a $(1/4)$-error probabilistic polynomial for OR_{X_0} of degree D.

Let $\mathbf{P}'_1, \ldots, \mathbf{P}'_s$ be $s = \log\log n$ independent instances of $\mathbf{P}'$ and let $\mathbf{Q} = 1 - \prod_{i \in [s]}(1 - \mathbf{P}'_i)$. Then, $\mathbf{Q}$ is an error $\frac{1}{4^s} = \frac{1}{\log^2 n}$ probabilistic polynomial for OR_n of degree at most $sD < \sqrt{\log n}/A\sqrt{\log\log n}$. In particular, there is a polynomial $q_0 \in \mathbb{R}[x_1, \ldots, x_n]$ of degree $d_0 < \sqrt{\log n}/A\sqrt{\log\log n}$ such that $q_0(0, 0, \ldots, 0) = 0$ and for $\varepsilon_0 = \frac{1}{\log^2 n}$ we have

$$\mathop{\mathbf{E}}_{i \in [(\log n)/2]}[\mathrm{Err}_i^{X_0}(q_0)] \leq \varepsilon_0$$

Define $n_0 = |X_0| = n$ and $\ell_0 = (\log n)/2$. By the above inequality, the polynomial q_0 is $(X_0, \ell_0, \varepsilon_0)$-good. Also define parameters $r = (d_0 \cdot \log^2 n)^{10d_0}$ and $p = \frac{1}{2^b}$ where $b \in \mathbb{N}$ is chosen so that $p \in [\frac{1}{2r^2}, \frac{1}{r^2}]$. Note that $r = n^{o(1)}$ and hence $p = \frac{1}{n^{o(1)}}$.

We now define a sequence of polynomials $q_1, q_2, \ldots, q_t$ such that:

- Each $q_i \in \mathbb{R}[X_i]$ where $X_i \subseteq X_0$ and has degree $d_i \geq 0$. Also, $|X_i| = n_i$ where $n_i \in [pn_{i-1}/2, 3pn_{i-1}/2]$. Further $\deg(q_i) = d_i < d_{i-1}$. The polynomial $q_i = q_{i-1}|_{\rho_i}$ for some restriction $\rho_i : X_{i-1} \to \{*, 0\}$.

- Each polynomial q_i is $(X_i, \ell_i, \varepsilon_i)$-good where $\ell_i = \ell_{i-1} - b$ and $\varepsilon_i = \varepsilon_{i-1} \cdot \exp\left(\frac{16b}{\log n}\right)$.
- $d_t = \deg(q_t) = 0$. That is, q_t is a constant polynomial.

Before we describe how to construct this sequence, let us see how it implies the desired contradiction. Note that since $d_i < d_{i-1}$ for each $i \geq 1$, the length t of the sequence is bounded by $d_0 < \sqrt{\log n}/A\sqrt{\log\log n}$.

We first make the following simple claim.

▶ **Claim 16.** *There is a large enough constant A in the definition of D above so that for each $i \in [t]$, $n_i \geq \sqrt{n}$, $\ell_i \geq \frac{\log n}{4}$, and $\varepsilon_i < \frac{1}{\log n}$.*

Proof. It can be checked that the following inequalities hold for a large enough choice of the constant A. Firstly,

$$n_i \geq n_t \geq n_0 \cdot (p/2)^t = n \cdot (d_0 \log n)^{-O(d_0^2)} \geq \sqrt{n}.$$

Also, note that $\ell_i = \ell_0 - bi \geq \ell_0 - bt = (\log n)/2 - O(d_0^2 \log\log n) \geq \frac{\log n}{4}$ and

$$\varepsilon_i = \varepsilon_0 \cdot \exp\left(\frac{16bi}{\log n}\right) \leq \varepsilon_0 \cdot \exp\left(\frac{16bt}{\log n}\right) = \frac{1}{\log^2 n} \cdot \exp\left(\frac{O(d_0^2 \log\log n)}{\log n}\right) < \frac{1}{\log n}. \qquad ◀$$

In particular, since q_t is $(X_t, \ell_t, \varepsilon_t)$-good, we must have

$$\mathrm{Err}_1^{X_t}(q_t) \leq \ell_t \mathop{\mathbf{E}}_{i \in [\ell_t]}[\mathrm{Err}_i^{X_t}(q_t)] < \varepsilon_t \ell_t < \frac{1}{2} \tag{1}$$

using the fact that $\ell_t \leq \ell_0 = (\log n)/2$ and $\varepsilon_t < \frac{1}{\log n}$.

Since $n_t \geq \sqrt{n}$, the function $\mathrm{OR}_{X_t}(x)$ evaluates to 1 under the distribution $\mu_1^{X_t} = \mathcal{U}_{X_t}$ with probability $1 - o(1)$. Thus, q_t must also evaluate to 1 on some input. However, since q_t is a constant polynomial, this implies that $q_t = 1$. But this implies that $q_t(0, 0, \ldots, 0) = 1$ as well, which leads to a contradiction, since q_t is obtained by setting some input bits of q_0 to 0 and $q_0(0, 0, \ldots, 0) = 0$ by our choice of q_0. This completes the proof of the theorem.

Now we describe how to obtain the sequence $q_1, \ldots, q_t$. More precisely, we describe how to obtain q_i from q_{i-1} assuming $d_{i-1} \geq 1$. Fix any $i \geq 1$ such that $d_{i-1} \geq 1$. We assume that the sequence $q_1, \ldots, q_{i-1}$ of polynomials constructed so far satisfy the above properties.

For brevity, let $q, X, m, d, \ell, \varepsilon$ denote $q_{i-1}, X_{i-1}, n_{i-1}, d_{i-1}, \ell_{i-1}, \varepsilon_{i-1}$ respectively.

We know that q is (X, ℓ, ε)-good. As we did in (1) for q_t, we can use this to show that $\mathrm{Err}_1^X(q) < \frac{1}{2}$ and since $\mathrm{OR}_X(x)$ takes the value 1 on an input $x \sim \mathcal{U}_X$ with probability $1 - o(1)$, we see that

$$\Pr_{x \sim \mathcal{U}_X}[q(x) = 1] \geq \frac{1}{2} - o(1) \geq \frac{1}{3}. \tag{2}$$

Lemma 14 then implies that there cannot r disjoint monomials of degree d in q. To see this, assume that there are indeed r many disjoint monomials of degree d in q. Then by Lemma 14, the probability that $q(x) - 1 = 0$ for a random $x \sim \mathcal{U}_X$ is at most

$$\begin{aligned} Bd^{4/3} r^{-\frac{1}{4d+1}} \sqrt{\log r} &\leq Bd_0^{4/3} r^{-\frac{1}{5d_0}} \sqrt{\log r} \\ &\leq Bd_0^{4/3} \cdot \frac{\sqrt{10 d_0 \log(d_0 \log^2 n)}}{d_0^2 \log^4 n} = o(1). \end{aligned}$$

This contradicts (2).

Hence, we know that q cannot be contain more than r many disjoint monomials of degree d. Let S be any maximal set of disjoint monomials appearing in q. Note that by definition, every monomial of degree d contains at least one variable from S and hence setting all the variables in S reduces the degree of the polynomial. The number of variables appearing in S is at most $d|S| \leq dr$.

We now choose a random restriction ρ with $*$-probability p as defined above and consider the polynomial $q|_\rho$. Define the following "bad" events:

- $\mathcal{E}_1(\rho)$ is the event that $|X_\rho| \not\in [pm/2, 3pm/2]$.
- $\mathcal{E}_2(\rho)$ is the event that some variable in S is not set to 0.
- $\mathcal{E}_3(\rho)$ is the event that $q|_\rho$ is not $(X_\rho, \ell', \varepsilon')$-good where $\ell' = \ell - b$ and $\varepsilon' = \varepsilon \cdot \exp\left(\frac{16b}{\log n}\right)$.

We claim that there is a ρ so that none of the bad events $\mathcal{E}_1(\rho), \mathcal{E}_2(\rho)$ or $\mathcal{E}_3(\rho)$ occur. This will imply that we can take $q_i = q|_\rho, X_i = X_\rho, \ell_i = \ell', \varepsilon_i = \varepsilon'$ and we will be done. So we only need to show that $\Pr_\rho[\mathcal{E}_1(\rho) \vee \mathcal{E}_2(\rho) \vee \mathcal{E}_3(\rho)] < 1$. This is done as follows.

- $\Pr_\rho[\mathcal{E}_1(\rho)]$: By Claim 16, we know that $m \geq \sqrt{n}$ and hence $\mathbf{E}_\rho[|X_\rho|] = pm = m \cdot \frac{1}{n^{o(1)}} \geq n^{1/4}$. Hence, by a Chernoff bound, the probability that $|X_\rho| \not\in [pm/2, 3pm/2]$ is bounded by $\exp(-\Omega(n^{1/4}))$.
- $\Pr_\rho[\mathcal{E}_2(\rho)]$: By a union bound over S, this probability is bounded by $p|S| \leq rd_0/r^2 < \frac{1}{\log n}$.
- $\Pr_\rho[\mathcal{E}_3(\rho)]$: By Observation 15, we know that for any i,

$$\mathop{\mathbf{E}}_{\rho}[\mathrm{Err}_i^{X_\rho}(q|_\rho)] = \mathrm{Err}_{i+b}^X(q).$$

Hence,

$$\mathop{\mathbf{E}}_{\rho}[\mathop{\mathbf{E}}_{i\in[\ell']}[\mathrm{Err}_i^{X_\rho}(q|_\rho)]] = \mathop{\mathbf{E}}_{i\in[\ell']}[\mathrm{Err}_{i+b}^X(q)] = \mathop{\mathbf{E}}_{i\in\{b+1,\ldots,b+\ell'\}}[\mathrm{Err}_i^X(q)] = \mathop{\mathbf{E}}_{i\in\{b+1,\ldots,\ell\}}[\mathrm{Err}_i^X(q)]. \tag{3}$$

We can bound the right hand side of the above equation by

$$\mathop{\mathbf{E}}_{i\in\{b+1,\ldots,\ell\}}[\mathrm{Err}_i^X(q)] \leq \frac{1}{(1-\frac{b}{\ell})} \mathop{\mathbf{E}}_{i\in[\ell]}[\mathrm{Err}_i^X(q)] \leq \frac{\varepsilon}{(1-\frac{b}{\ell})}$$

where the final inequality follows from the fact that q is (X, ℓ, ε)-good. Further, by Claim 16, we know that $\ell \geq \frac{\log n}{4} \gg b$, and hence we can bound the above as follows.

$$\mathop{\mathbf{E}}_{i\in\{b+1,\ldots,\ell\}}[\mathrm{Err}_i^X(q)] \leq \frac{\varepsilon}{(1-\frac{b}{\ell})} \leq \varepsilon \cdot (1 + \frac{2b}{\ell}) \leq \varepsilon \cdot (1 + \frac{8b}{\log n}).$$

Plugging the above bound into (3), we obtain

$$\mathop{\mathbf{E}}_{\rho}[\mathop{\mathbf{E}}_{i\in[\ell']}[\mathrm{Err}_i^{X_\rho}(q|_\rho)]] \leq \varepsilon \cdot (1 + \frac{8b}{\log n}) \leq \varepsilon \cdot \exp\left(\frac{8b}{\log n}\right).$$

By Markov's inequality,

$$\Pr_{\rho}[\mathop{\mathbf{E}}_{i\in[\ell']}[\mathrm{Err}_i^{X_\rho}(q|_\rho)] > \varepsilon \cdot \exp\left(\frac{16b}{\log n}\right)] \leq \exp\left(-\frac{8b}{\log n}\right) = 1 - \Omega(\frac{b}{\log n}) \leq 1 - \frac{2}{\log n}.$$

Thus, $\Pr_\rho[\mathcal{E}_3(\rho)] \leq 1 - \frac{2}{\log n}$.

By a union bound, we have

$$\Pr_{\rho}[\mathcal{E}_1(\rho) \vee \mathcal{E}_2(\rho) \vee \mathcal{E}_3(\rho)] \leq \exp(-\Omega(n^{1/4})) + \frac{1}{\log n} + 1 - \frac{2}{\log n} < 1.$$

Acknowledgements. We thank Swagato Sanyal and Madhu Sudan for encouragement and useful discussions which greatly simplified our proofs. We thank Avishay Tal for his generous feedback and comments and also for showing us the improvement in seedlength mentioned in Remark 2.2. We also thank Paul Beame and Xin Yang for pointing out that a change in parameters results in a quantitative improvement in the lower bound obtained in Theorem 13.

References

1 Miklós Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computations. In *Proc. 16th ACM Symp. on Theory of Computing (STOC)*, pages 471–474, 1984. doi:10.1145/800057.808715.

2 Richard Beigel, Nick Reingold, and Daniel A. Spielman. The perceptron strikes back. In *Proc. 6th IEEE Conf. on Structure in Complexity Theory*, pages 286–291, 1991. doi:10.1109/SCT.1991.160270.

3 Mark Braverman. Polylogarithmic independence fools AC^0 circuits. *J. ACM*, 57(5), 2010. (Preliminary version in *24th IEEE Conference on Computational Complexity*, 2009). doi:10.1145/1754399.1754401.

4 Kevin P. Costello, Terence Tao, and Van Vu. Random symmetric matrices are almost surely nonsingular. *Duke Math. J.*, 135(2):395–413, 2006. arXiv:math/0505156, doi:10.1215/S0012-7094-06-13527-5.

5 Paul Erdős. On a lemma of Littlewood and Offord. *Bull. Amer. Math. Soc.*, 51(12):898–902, 1945. doi:10.1090/S0002-9904-1945-08454-7.

6 Johan Håstad. Almost optimal lower bounds for small depth circuits. In Silvio Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 143–170. JAI Press, Greenwich, Connecticut, 1989. (Preliminary version in *18th STOC* 1986). URL: http://www.csc.kth.se/~johanh/largesmalldepth.pdf.

7 Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM J. Comput.*, 43(5):1699–1708, 2014. doi:10.1137/120897432.

8 Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *Proc. 23rd Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 961–972, 2012. URL: http://arxiv.org/abs/1107.3127.

9 Swastik Kopparty and Srikanth Srinivasan. Certifying polynomials for AC^0(parity) circuits, with applications. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *Proc. 32nd IARCS Annual Conf. on Foundations of Software Tech. and Theoretical Comp. Science (FSTTCS)*, volume 18 of *LIPIcs*, pages 36–47. Schloss Dagstuhl, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.36.

10 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993. (Preliminary version in *30th FOCS*, 1989). doi:10.1145/174130.174138.

11 John Edensor Littlewood and A. Cyril Offord. On the number of real roots of a random algebraic equation. *J. London Math. Soc.*, s1-13(4):288–295, 1938. doi:10.1112/jlms/s1-13.4.288.

12 Michael Luby and Boban Velickovic. On deterministic approximation of DNF. *Algorithmica*, 16(4/5):415–433, 1996. (Preliminary version in *23rd STOC*, 1991). doi:10.1007/BF01940873.

13 Raghu Meka, Oanh Nguyen, and Van Vu. Anti-concentration for polynomials of independent random variables. arXiv 1507.00829, 2015. URL: https://arxiv.org/abs/1507.00829.

14 Noam Nisan and Avi Wigderson. Hardness vs. Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, October 1994. (Preliminary version in *29th FOCS*, 1988). `doi:10.1016/S0022-0000(05)80043-1`.

15 Ryan O'Donnell. *Analysis of Boolean Functions.* Cambridge University Press, 2014. URL: `http://analysisofbooleanfunctions.org/`, `doi:10.1017/CBO9781139814782`.

16 Igor Carboni Oliveira and Rahul Santhanam. Majority is incompressible by $AC^0[p]$ circuits. In *Proc.* 30*th Computational Complexity Conf.*, pages 124–157, 2015. `doi:10.4230/LIPIcs.CCC.2015.124`.

17 Alexander A. Razborov. Нжние оценки размера схем ограниченной глубины в полном базисе, содержащем функцию логического сложения (Russian) [Lower bounds on the size of bounded depth circuits over a complete basis with logical addition]. *Mathematicheskie Zametki*, 41(4):598–607, 1987. (English translation in *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987). URL: `http://mi.mathnet.ru/eng/mz4883`, `doi:10.1007/BF01137685`.

18 Alexander A. Razborov and Emanuele Viola. Real advantage. *ACM T. Comput. Theory*, 5(4):17, 2013. `doi:10.1145/2540089`.

19 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proc.* 19*th ACM Symp. on Theory of Computing (STOC)*, pages 77–82, 1987. `doi:10.1145/28395.28404`.

20 Avishay Tal. Tight bounds on the fourier spectrum of AC^0. Technical Report TR14-174, Elect. Colloq. on Comput. Complexity (ECCC), 2014.

21 Jun Tarui. Probablistic polynomials, AC^0 functions, and the polynomial-time hierarchy. *Theoret. Comput. Sci.*, 113(1):167–183, 1993. (Preliminary Version in *8th STACS*, 1991). `doi:10.1016/0304-3975(93)90214-E`.

22 Seinosuke Toda and Mitsunori Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 21(2):316–328, 1992. (Preliminary version in *6th Structure in Complexity Theory Conference*, 1991). `doi:10.1137/0221023`.

23 Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of AC^0. In *Proc.* 28*th IEEE Conf. on Computational Complexity*, pages 242–247, 2013. `doi:10.1109/CCC.2013.32`.

24 Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *Proc.* 46*th ACM Symp. on Theory of Computing (STOC)*, pages 194–202, 2014. `arXiv:1401.2444`, `doi:10.1145/2591796.2591858`.

A The limitations of the Nisan Wigderson paradigm

In this section, we show that the general hardness-to-randomness tradeoff of Nisan and Wigderson [14] does not yield a PRG with optimal seedlength as a function of ε *given our current knowledge* of circuit lower bounds.

We start by describing the meta-result of Nisan and Wigderson [14] that allows us to convert any sufficiently hard-to-compute function for a class of circuits to a PRG for a slightly weaker class of circuits. The result is true in greater generality than we describe here but to keep things concrete, we stick to the setting of $\mathrm{AC}^0(s,d)$.

We say that a function $f:\{0,1\}^r \to \{0,1\}$ is (s,d,ε)-hard if given any circuit C from $\mathrm{AC}^0(s,d)$ of size s, we have

$$\Pr_{x\in\{0,1\}^r}[C(x)=f(x)] \leq \frac{1}{2}+\varepsilon.$$

For non-negative integers m,r,ℓ,s, we say that a family $\mathcal{F}\subseteq\binom{[m]}{r}$, we say that $\mathcal{F}$ is an (m,r,ℓ,s) design if $|\mathcal{F}|=s$ and for any distinct $S,T\in\mathcal{F}$, we have $|S\cap T|\leq\ell$.

Nisan and Wigderson [14] show the following.

▶ **Theorem 17** ([14]). *Let $m, r, \ell, s \in \mathbb{N}$ be positive parameters such that $m \geq r \geq \ell$. Given an explicit $f : \{0,1\}^r \to \{0,1\}$ that is $(s \cdot 2^\ell, d+1, \varepsilon/s)$-hard and an explicit (m, r, ℓ, s)-design, we can construct an explicit PRG $G : \{0,1\}^m \to \{0,1\}^s$ that fools circuits from $\mathrm{AC}^0(s,d)$ with error at most ε.*

To use this theorem, we need a hard function for circuits in AC^0. The best such result known currently is the following due to Impagliazzo, Matthews, and Paturi [8] (see also Håstad [7]).

▶ **Theorem 18.** *Let $d \geq 1$ be a constant. The Parity function on r is bits is (s_1, d_1, δ)-hard if $r \geq A(\log s_1)^{d_1 - 1} \cdot \log(1/\delta)$ for some constant $A > 0$ depending on d.*

Thus, if we want to apply Theorem 17 alongside the lower bound given by Theorem 18 to construct PRGs that ε-fool $\mathrm{AC}^0(s,d)$, then we need

$$r \geq A(\log s + \ell)^d \cdot \log(s/\varepsilon) \geq A(\log s + \ell)^d \cdot \log(1/\varepsilon) \tag{4}$$

for some constant $A > 0$ depending on d.

Further, to construct an (m, r, ℓ, s)-design, we claim that we further need

$$m \geq \min\{r^2/2\ell, s\}. \tag{5}$$

We justify (5) below, but first we use it to prove that the Nisan-Wigderson paradigm cannot be used to obtain seedlength optimal in terms of ε for a large range of ε.

We assume that $\varepsilon \geq \exp(-s^{1/4})$ (the same proof works as long as $\varepsilon \geq \exp(-s^{\frac{1}{2}-\Omega(1)})$). In this setting, we show that $m \geq B(\log s)^{2d-1} \cdot (\log(1/\varepsilon))^2$ for some constant B depending on d.

To see this, note that if $m \geq s$, then trivially we have $(\log s)^{2d-1} \cdot (\log 1/\varepsilon)^2 \leq s^{\frac{1}{2}+o(1)} < s \leq m$. So we assume that $m < s$.

In this case, (5) tells us that $m \geq r^2/2\ell$, which yields

$$\begin{aligned} m &\geq \frac{r^2}{2\ell} \geq \frac{A^2(\log s + \ell)^{2d} \cdot (\log 1/\varepsilon)^2}{2\ell} \\ &\geq \frac{A^2(\log s)^{2d-1}\ell(\log 1/\varepsilon)^2}{2\ell} = \Omega(A^2(\log s)^{2d-1} \cdot (\log(1/\varepsilon))^2) \end{aligned}$$

as required.

The inequality (5) is a standard combinatorial fact and can be found in many standard textbooks. For completeness, here is a simple proof using inclusion-exclusion.

Note that if $s \leq r$, then we immediately have $m \geq r \geq s$ and (5) is proved. So assume that $s > r$ and in particular given any (m, r, ℓ, s)-design $\mathcal{F}$, we can choose $t = r/\ell$ sets $T_1, \ldots, T_t$ from $\mathcal{F}$. By inclusion-exclusion, we have

$$\begin{aligned} m &\geq |\bigcup_{i \in [t]} T_i| \geq \sum_i |T_i| - \sum_{i<j} |T_i \cap T_j| \\ &\geq rt - \frac{t^2}{2} \cdot \ell \geq \frac{r^2}{2\ell} \end{aligned}$$

which concludes the proof of (5).

On the Structure of Quintic Polynomials

Pooya Hatami[*]

Institute for Advanced Study, Princeton, NJ, USA
`pooyahat@math.ias.edu`

Abstract

We study the structure of bounded degree polynomials over finite fields. Haramaty and Shpilka [STOC 2010] showed that biased degree three or four polynomials admit a strong structural property. We confirm that this is the case for degree five polynomials also. Let $\mathbb{F} = \mathbb{F}_q$ be a prime field. Suppose $f : \mathbb{F}^n \to \mathbb{F}$ is a degree five polynomial with $\text{bias}(f) = \delta$. We prove the following two structural properties for such f.

1. We have $f = \sum_{i=1}^{c} G_i H_i + Q$, where G_i and H_is are nonconstant polynomials satisfying $\deg(G_i) + \deg(H_i) \leqslant 5$ and Q is a degree $\leqslant 4$ polynomial. Moreover, c does not depend on n.
2. There exists an $\Omega_{\delta,q}(n)$ dimensional affine subspace $V \subseteq \mathbb{F}^n$ such that $f|_V$ is a constant.

Cohen and Tal [Random 2015] proved that biased polynomials of degree at most four are constant on a subspace of dimension $\Omega(n)$. Item [2.] extends this to degree five polynomials. A corollary to Item [2.] is that any degree five affine disperser for dimension k is also an affine extractor for dimension $O(k)$. We note that Item [2.] cannot hold for degrees six or higher.

We obtain our results for degree five polynomials as a special case of structure theorems that we prove for biased degree d polynomials when $d < |\mathbb{F}| + 4$. While the $d < |\mathbb{F}| + 4$ assumption seems very restrictive, we note that prior to our work such structure theorems were only known for $d < |\mathbb{F}|$ by Green and Tao [Contrib. Discrete Math. 2009] and Bhowmick and Lovett [arXiv:1506.02047]. Using algorithmic regularity lemmas for polynomials developed by Bhattacharyya, et. al. [SODA 2015], we show that whenever such a strong structure exists, it can be found algorithmically in time polynomial in n.

1998 ACM Subject Classification G.2.1 Combinatorics

Keywords and phrases Higher-order Fourier analysis, Structure Theorem, Polynomials, Regularity lemmas

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.33

1 Introduction

Let $\mathbb{F} = \mathbb{F}_q$ be a prime field. The bias of a function $f : \mathbb{F}^n \to \mathbb{F}$ is defined as

$$\text{bias}(f) := \left| \mathop{\mathbf{E}}_{x \in \mathbb{F}^n} \left[\omega^{f(x)} \right] \right|,$$

where $\omega = e^{2\pi i/|\mathbb{F}|}$, is a complex primitive root of unity of order $|\mathbb{F}|$. The smaller the bias of a function, the more uniformly f is distributed over $\mathbb{F}$, thus a random function has negligible bias. This remains true, if f is a random degree d polynomial for a fixed degree $d > 0$. Thus bias can be thought of as a notion of pseudorandomness for polynomials, and as often lack of pseudorandomness implies structure, one may ask whether every biased degree d polynomial admits strong structural properties. Green and Tao [7] (in the case when $d < |\mathbb{F}|$) and later Kaufman and Lovett [11] (in the general case) proved this heuristic to be true by showing that

[*] Supported by the National Science Foundation grant No. CCF-1412958.

every biased degree d polynomial is determined by a few lower degree polynomials. Formally, these results state that for a degree d polynomial f, there is a constant $c \leqslant c(d, \text{bias}(f), |\mathbb{F}|)$, degree $\leqslant d-1$ polynomials $Q_1, \dots, Q_c$ and a function $\Gamma : \mathbb{F}^c \to \mathbb{F}$, such that

$$f = \Gamma(Q_1, \dots, Q_c). \tag{1}$$

Note that here crucially c does not depend on the dimension n, meaning that for large n, it is very unlikely for a typical polynomial to be biased. Recently, Bhowmick and Lovett [3] proved that the dependence of the number of terms in (1) on $|\mathbb{F}|$ can be removed, in other words biased polynomials are very rare even when the field size is allowed to grow with n. These structure theorems for biased polynomials have had several important applications. For example they were used by Kaufman and Lovett [11] to give interesting worst case to average case reductions, and by Tao and Ziegler [16] in their proof of the inverse theorem for Gowers norms over finite fields. Such structure theorems have played an important role in determining the weight distribution and list decoding radius of Reed-Muller codes [12, 4, 3]. They were also used by Cohen and Tal [5] to show that any degree d affine disperser over a prime field is also an affine extractor with related parameters.

There are however two drawbacks to the structure theorems proved in [7, 11]. Firstly, the constant c has very bad dependence on δ which is due to the use of regularity lemmas for polynomials. Secondly, there is no restrictions on the function Γ obtained in (1), in particular there is nothing stopping it from being of degree c. In the special case of quadratic polynomials better bounds and structural properties follow from the following well-known theorem.

▶ **Theorem 1** (Structure of quadratic polynomials [13]). *For every quadratic polynomial $f : \mathbb{F}^n \to \mathbb{F}$ over a prime field $\mathbb{F}$, there exists an invertible linear map T, a linear polynomial ℓ, and field elements $\alpha_1, \dots, \alpha_n$ such that*

- *If $|\mathbb{F}| = 2$, then $(f \circ T)(x) = \sum_{i=1}^{\lfloor n/2 \rfloor} \alpha_i x_{2i-1} x_{2i} + \ell(x)$.*
- *If $|\mathbb{F}|$ is odd, then $(f \circ T)(x) = \sum_{i=1}^{n} \alpha_i x_i^2 + \ell(x)$.*

It easily follows that every quadratic polynomial f, can be written in the form $\sum_{i=1}^{2\log(1/\text{bias}(f))} \ell_i \ell'_i + \ell''$ where ℓ_i, ℓ'_is and ℓ'' are linear polynomials. This is a very strong structural property, moreover the dependence of the number of the terms on bias(f) is optimal. Haramaty and Shpilka [8] studied the structure of biased cubic and quartic polynomials and proved the following two theorems.

▶ **Theorem 2** (Biased cubic polynomials [8]). *Let $f : \mathbb{F}^n \to \mathbb{F}$ be a cubic polynomial such that* bias$(f) = \delta > 0$. *Then there exist $c_1 = O(\log(1/\delta))$, $c_2 = O\left(\log^4(1/\delta)\right)$, quadratic polynomials $Q_1, ..., Q_{c_1} : \mathbb{F}^n \to \mathbb{F}$, linear functions $\ell_1, ..., \ell_{c_1}, \ell'_1, ..., \ell'_{c_2} : \mathbb{F}^n \to \mathbb{F}$ and a cubic polynomial $\Gamma : \mathbb{F}^{c_2} \to \mathbb{F}$ such that*

$$f = \sum_{i=1}^{c_1} \ell_i Q_i + \Gamma\left(\ell'_1, \dots, \ell'_{c_2}\right).$$

▶ **Theorem 3** (Biased quartic polynomials [8]). *Let $f : \mathbb{F}^n \to \mathbb{F}$ be a quartic polynomial such that* bias$(f) = \delta$. *There exist $c = \text{poly}(|\mathbb{F}|/\delta)$ and polynomials $\{\ell_i, Q_i, Q'_i, G_i\}_{i \in [c]}$, where the ℓ_is are linear, Q_i, Q'_is are quadratic, and G_i's are cubic polynomials, such that*

$$f = \sum_{i=1}^{c} \ell_i G_i + \sum_{i=1}^{c} Q_i Q'_i.$$

In the high characteristic regime when $d = \deg(f) < |\mathbb{F}|$, Green and Tao [7] showed that such a strong structure theorem holds, with a dependence that is really large in terms of bias. More precisely, if $d < |\mathbb{F}|$, then every degree d polynomial f, with $\text{bias}(f) \geqslant \delta$ can be written in the form $f = \sum_{i=1}^{c(\delta,\mathbb{F},d)} G_i H_i + Q$, where G_i and H_is are nonconstant polynomials satisfying $\deg(G_i) + \deg(H_i) \leqslant d$, and Q is a degree $\leqslant d-1$ polynomial. Recently, Bhowmick and Lovett [3] have proved that one can remove the dependence of c on $|\mathbb{F}|$. However, in the low characteristic case, i.e. when $|\mathbb{F}|$ can be smaller than d, the only general results before this work are Theorems 2 and 3.

1.1 Our results

Suppose that $\mathbb{F} = \mathbb{F}_q$ is a prime field. In this work we are interested in the case when q is a fixed prime, as the case of large q is addressed by a recent work of [3]. When the characteristic of $\mathbb{F}$ can be small, namely when $|\mathbb{F}| \leqslant 5$, it was not known whether a degree five biased polynomial admits a strong structure in the sense of Theorems 2 and 3. Moreover, the techniques from [8] seem to break down.

Quintic polynomials

We combine ideas from [8] with arguments from polynomial regularity and prove such a structure theorem for quintic polynomials.

▶ **Theorem 4** (Biased quintic polynomials I). *Suppose $f : \mathbb{F}^n \to \mathbb{F}$ is a degree five polynomial with $\text{bias}(f) = \delta$. There exist $c_4 \leqslant c(\delta, |\mathbb{F}|)$, nonconstant polynomials $G_1, ..., G_c, H_1, ..., H_c$ and a polynomial Q such that the following holds.*

- $f = \sum_{i=1}^{c} G_i H_i + Q$.
- *For every $i \in [c]$, $\deg(G_i) + \deg(H_i) \leqslant 5$.*
- $\deg(Q) \leqslant 4$.

Note that c_4 has no dependence on n. We also prove that every biased quintic polynomial is constant on an affine subspace of dimension $\Omega(n)$.

▶ **Theorem 5** (Biased quintic polynomials II). *Suppose $f : \mathbb{F}^n \to \mathbb{F}$ is a degree five polynomial with $\text{bias}(f) = \delta$. There exists an affine subspace V of dimension $\Omega(n)$ such that $f|_V$ is constant.*

Theorem 5 was previously only known for degrees $\leqslant 4$. The case of quadratics when $\mathbb{F} = \mathbb{F}_2$ is Dickson's theorem [6], and the case of general $\mathbb{F}$ and $d \leqslant 4$ was proved recently by Cohen and Tal [5] building on Theorem 2 and Theorem 3. We also remark that the degree five is the largest degree that such a bound can hold. To see this, assume for example that $d = 6$ and $\mathbb{F} = \mathbb{F}_2$, and construct a degree 6 polynomial $f = G(x_1, ..., x_n) \cdot H(x_1, \ldots, x_n)$ by picking two random cubic polynomials G and H. One observes that f has large bias as $\mathbf{Pr}(f = 0) = 3/4 + o(1)$, however, f will not vanish over any subspace of dimension $\Omega(n^{1/2})$. Theorem 5 has the following immediate corollary.

▶ **Corollary 6.** *Suppose $f : \mathbb{F}^n \to \mathbb{F}$ is a degree five affine disperser for dimension k. Then f is also an affine extractor of dimension $O(k)$.*

We refer to [5] where affine dispersers and extractors and the relations between them are discussed.

Degree d polynomials, with $d < |\mathbb{F}| + 4$

We in fact prove a strong structure theorem for biased degree d polynomials when $d < |\mathbb{F}|+4$, from which Theorem 4 follows immediately.

▶ **Theorem 7** (Biased degree d polynomials I (when $d < |\mathbb{F}| + 4$)). *Suppose $0 < d$ and $\mathbb{F}$ is a prime field satisfying $d < |\mathbb{F}| + 4$. Let $f : \mathbb{F}^n \to \mathbb{F}$ be a degree d polynomial with* $\text{bias}(f) = \delta$. *There exists $c_7 \leqslant c(\delta, d, |\mathbb{F}|)$, nonconstant polynomials $G_1, ..., G_c, H_1, ..., H_c$ and a polynomial Q such that the following hold.*

- $f = \sum_{i=1}^{c} G_i H_i + Q$.
- *For every $i \in [c]$,* $\deg(G_i) + \deg(H_i) \leqslant d$.
- $\deg(Q) \leqslant d - 1$.

We also prove a general version of Theorem 5 when $d < |\mathbb{F}| + 4$.

▶ **Theorem 8** (Biased degree d polynomials II (when $d < |\mathbb{F}| + 4$)). *Suppose $0 < d$ and $\mathbb{F}$ is a prime field satisfying $d < |\mathbb{F}| + 4$. Let $f : \mathbb{F}^n \to \mathbb{F}$ be a degree d polynomial with* $\text{bias}(f) = \delta$. *There exists an affine subspace V of dimension $\Omega_{d,\delta}(n^{1/\lfloor \frac{d-2}{2} \rfloor})$ such that $f|_V$ is a constant.*

Cohen and Tal [5] recently showed that any degree d biased polynomial is constant on an $\Omega_\delta(n^{1/(d-1)})$ dimensional affine subspace. Theorem 8 improves on this by a quadratic factor, when $d < |\mathbb{F}| + 4$.

Our results for quintic polynomials follow immediately.

Proof of Theorem 4 and Theorem 5. Theorem 4 and Theorem 5 follow curiously as special cases of Theorem 7 and Theorem 8 as $|\mathbb{F}| \geqslant 2$ and $5 < 2 + 4$. ◀

Algorithmic aspects

Using a result of Bhattacharyya, et. al. [2] who gave an algorithm for finding prescribed decompositions of polynomials, we show that whenever such a strong structure exists, it can be found algorithmically in time polynomial in n. Combined with Theorem 7, we obtain the following algorithmic structure theorem.

▶ **Theorem 9.** *Suppose $\delta > 0$, $d > 0$ are given, and let $\mathbb{F}$ be a prime field satisfying $d < |\mathbb{F}|+4$. There is a deterministic algorithm that runs in time $O(n^{O(d)})$ and given as input a degree d polynomial $f : \mathbb{F}^n \to \mathbb{F}$ satisfying* $\text{bias}(f) = \delta$, *outputs a number $c \leqslant c(\delta, |\mathbb{F}|, d)$, a collection of degree $\leqslant d - 1$ polynomials $G_1, ..., G_c, H_1, ..., H_c : \mathbb{F}^n \to \mathbb{F}$ and a polynomial $Q : \mathbb{F}^n \to \mathbb{F}$, such that*

- $f = \sum_{i=1}^{c} G_i H_i + Q$.
- *For every $i \in [c]$,* $\deg(G_i) + \deg(H_i) \leqslant d$.
- $\deg(Q) \leqslant d - 1$.

1.2 Overview of Proofs

As we saw above, Theorems 4 and 5 are immediate consequences of Theorems 7 and 8.

Proof overview of Theorem 7. Given a degree d biased polynomial $f : \mathbb{F}^n \to \mathbb{F}$, using an additive combinatorial argument we find a bounded index subspace restricted to which all the first degree partial derivatives of f are biased. We observe that since f was biased, it must be biased in some coset of this subspace also, and hence by a result of [11] it must be a function of a constant number of its derivatives. As each derivative of f is a degree $\leqslant d - 1$ and biased, we again invoke the "bias implies low-rank" result of [11] for these lower-degree polynomials in order to rewrite f as a function of a constant number of degree $d-2$ polynomials. We finally show that under the assumption that $d < |\mathbb{F}|+4$, we can "regularize" the resulting polynomials to a "regular" collection of polynomials from which the structure theorem can be deduced.

Proof overview of Theorem 8. Following the proof of Theorem 7, we pick an affine subspace W of constant codimension restricted to which f has the nice structure $f|_W = \sum G_i H_i + M$, where G_i, H_i, M are all of degrees $\leqslant d-2$. We moreover know that G_i, H_i, M are functions of a regular set of polynomials of degree $\leqslant d-2$. We argue by looking at the higher-order Fourier expansion of f that M must be a constant field element and since for each i, $\min\{\deg(G_i), \deg(H_i)\} \leqslant \lfloor \frac{d}{2} \rfloor$, using a result of Cohen and Tal [5] we can restrict to a subspace of dimension $\Omega(n^{1/\lfloor \frac{d-2}{2} \rfloor})$ making f a constant.

Organization

In Section 2 we present the basic tools from higher-order Fourier analysis. In Section 3 we discuss useful properties of a pseudorandom collection of polynomials. Theorem 7 is proved in Section 4.1, and Theorem 8 is proved in Section 4.2. We discuss the algorithmic aspects in Section 5. We end with a discussion of future directions in Section 6.

Notation

Let $\mathbb{D} = \{z \in \mathbb{C} : |z| \leqslant 1\}$ be the unit disk in the complex plane. Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$. Suppose that $\mathbb{F} = \mathbb{F}_q$ is a finite prime field, let $e_{\mathbb{F}} : \mathbb{F} \to \mathbb{D}$ denote the function $e_{\mathbb{F}}(x) := e^{\frac{2\pi i x}{|\mathbb{F}|}}$, and let $e : \mathbb{T} \to \mathbb{D}$ denote the function $e(x) := e^{2\pi i x}$. For functions $f, g : \mathbb{F}^n \to \mathbb{C}$, define

$$\langle f, g \rangle := \frac{1}{|\mathbb{F}|^n} \sum_{x \in \mathbb{F}^n} f(x)\overline{g(x)}.$$

For an integer a, denote by $[a] := \{1, \ldots, a\}$.

2 Preliminary results from higher-order Fourier analysis

2.1 Nonclassical Polynomials

Let $d \geqslant 0$ be an integer. It is well-known that for functions $P : \mathbb{F}^n \to \mathbb{F}$, a polynomial of degree $\leqslant d$ can be defined in two different ways. We say that P is a polynomial of degree $\leqslant d$ if it can be written as

$$P(x_1, ..., x_n) = \sum_{\substack{i_1,\ldots,i_n \geqslant 0 \\ i_1+\cdots+i_n \leqslant d}} c_{i_1,\ldots,i_n} x_1^{i_1} \cdots x_n^{i_n},$$

with coefficients $c_{i_1,\ldots,i_n} \in \mathbb{F}$. This can be thought of as a global definition for polynomials in $\mathbb{F}[x_1, \ldots, x_n]$. The local definition of a polynomial uses the notion of additive directional derivatives.

▶ **Definition 10** (Polynomials over finite fields (local definition))**.** Suppose that G is an abelian group. For an integer $d \geqslant 0$, a function $P : \mathbb{F}^n \to G$ is said to be a *polynomial of degree* $\leqslant d$ if for all $y_1, \ldots, y_{d+1}, x \in \mathbb{F}^n$, it holds that

$$(D_{y_1} \cdots D_{y_{d+1}} P)(x) = 0,$$

where $D_yP(x) = P(x+y) - P(x)$ is the additive derivative of P with direction y evaluated at x. The *degree* of P is the smallest d for which the above holds.

It follows simply from the definition that for any direction $y \in \mathbb{F}^n$, $deg(D_yP) < deg(P)$. In the "classical" case of polynomials $P : \mathbb{F}^n \to \mathbb{F}$, i.e. $G = \mathbb{F}$, it is a well-known fact that the global and local definitions coincide. However, the situation is different when G is allowed to be other groups. For example when the range of P is $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, it turns out that the global definition must be refined to the "nonclassical polynomials". This phenomenon was noted by Tao and Ziegler [16] in the study of Gowers norms.

Nonclassical polynomials arise when studying functions $P : \mathbb{F}^n \to \mathbb{T}$ and their exponents $f = e(P) : \mathbb{F}^n \to \mathbb{C}$.

▶ **Definition 11** (Nonclassical Polynomials). For an integer $d \geqslant 0$, a function $P : \mathbb{F}^n \to \mathbb{T}$ is said to be a *nonclassical polynomial of degree* $\leqslant d$ (or simply a *polynomial of degree* $\leqslant d$) if for all $y_1, \ldots, y_{d+1}, x \in \mathbb{F}^n$, it holds that

$$(D_{y_1} \cdots D_{y_{d+1}} P)(x) = 0. \tag{2}$$

The *degree* of P is the smallest d for which the above holds. A function $P : \mathbb{F}^n \to \mathbb{T}$ is said to be a *classical polynomial of degree* $\leqslant d$ if it is a nonclassical polynomial of degree $\leqslant d$ whose image is contained in $\frac{1}{q}\mathbb{Z}/\mathbb{Z}$.

Denote by $\mathrm{poly}(\mathbb{F}^n \to \mathbb{T})$, $\mathrm{poly}_d(\mathbb{F}^n \to \mathbb{T})$ and $\mathrm{poly}_{\leqslant d}(\mathbb{F}^n \to \mathbb{T})$, the set of all nonclassical polynomials over $\mathbb{F}^n$, all nonclassical polynomials of degree d and all nonclassical polynomials of degree $\leqslant d$ respectively.

From this point on by a polynomial we always mean a nonclassical polynomial, and we will make it clear when we talk about classical polynomials.

The following lemma of Tao and Ziegler [16] shows that a classical polynomial P of degree d must always be of the form $x \mapsto \frac{|Q(x)|}{q}$, where $Q : \mathbb{F}^n \to \mathbb{F}$ is a polynomial (in the usual sense) of degree d, and $|\cdot|$ is the standard map from $\mathbb{F}$ to $\{0, 1, \ldots, q-1\}$. This lemma also characterizes the structure of (nonclassical) polynomials.

▶ **Lemma 12** (Lemma 1.7 in [16]). *A function $P : \mathbb{F}^n \to \mathbb{T}$ is a polynomial of degree $\leqslant d$ if and only if P can be represented as*

$$P(x_1, \ldots, x_n) = \alpha + \sum_{\substack{0 \leqslant d_1, \ldots, d_n < q; k \geqslant 0: \\ 0 < \sum_i d_i \leqslant d - k(q-1)}} \frac{c_{d_1, \ldots, d_n, k} |x_1|^{d_1} \cdots |x_n|^{d_n}}{q^{k+1}} \mod 1,$$

for a unique choice of $c_{d_1, \ldots, d_n, k} \in \{0, 1, \ldots, q-1\}$ and $\alpha \in \mathbb{T}$. The element α is called the shift *of P, and the largest integer k such that there exist $d_1, \ldots, d_n$ for which $c_{d_1, \ldots, d_n, k} \neq 0$ is called the* depth *of P. A depth-k polynomial P takes values in an affine shift of the subgroup $\mathbb{U}_{k+1} := \frac{1}{q^{k+1}}\mathbb{Z}/\mathbb{Z}$. Classical polynomials correspond to polynomials with 0 shift and 0 depth.*

For convenience of exposition, henceforth we will assume that the shifts of all polynomials are zero. This can be done without affecting any of the results presented in this text. Under this assumption, all polynomials of depth k take values in $\mathbb{U}_{k+1}$.

2.2 Rank, Regularity, and Other Notions of Uniformity

The rank of a polynomial is a notion of its complexity according to lower degree polynomials.

▶ **Definition 13** (Rank of a polynomial). Given a polynomial $P : \mathbb{F}^n \to \mathbb{T}$ and an integer $d \geqslant 1$, the *d-rank* of P, denoted $\mathsf{rank}_d(P)$, is defined to be the smallest integer r such that there exist polynomials $Q_1, \ldots, Q_r : \mathbb{F}^n \to \mathbb{T}$ of degree $\leqslant d-1$ and a function $\Gamma : \mathbb{T}^r \to \mathbb{T}$ satisfying $P(x) = \Gamma(Q_1(x), \ldots, Q_r(x))$. If $d = 1$, then 1-rank is defined to be ∞ if P is non-constant and 0 otherwise.

The *rank* of a polynomial $P : \mathbb{F}^n \to \mathbb{T}$ is its $\deg(P)$-rank. We say that P is r-regular if $\mathsf{rank}(P) \geqslant r$.

Note that for an integer $\lambda \in [1, q-1]$, $\text{rank}(P) = \text{rank}(\lambda P)$. In this article we are interested in obtaining a structure theorem for biased classical polynomials that does not involve nonclassical polynomials. Motivated by this, we define two other notions of rank.

▶ **Definition 14** (Classical rank of a polynomial). Given a classical polynomial $P : \mathbb{F}^n \to \mathbb{F}$ and an integer $d \geqslant 1$, the classical d-rank of P, denoted by $\text{crank}_d(P)$, is defined similarly to Definition 13 with the extra restriction that $Q_1, ..., Q_r : \mathbb{F}^n \to \mathbb{F}$ are classical polynomials.

The *classical rank* of a classical polynomial $P : \mathbb{F}^n \to \mathbb{F}$ is its classical $\deg(P)$-rank. We say that P is classical r-regular if $\mathsf{crank}(P) \geqslant r$.

▶ Remark. For a nonconstant affine-linear polynomial $P(x)$, $\text{rank}(P) = \text{crank}(P) = \infty$ and for a constant function $Q(x)$, $\text{rank}(Q) = 0$.

▶ Remark. It is important to note that Definition 13 and Definition 14 are not equivalent. To see this, note that, as proved in [16] and [14], the degree 4 symmetric polynomial $S_4 := \sum_{i<j<k<\ell} x_i x_j x_k x_\ell$ has negligible correlation with any degree $\leqslant 3$ classical polynomial. A simple Fourier analytic argument implies that $\text{crank}(S_4) = \omega(1)$, i.e. $\lim_{n\to\infty} \text{crank}(S_4(x_1, ..., x_n)) = \infty$. However, S_4 turns out to have large *Gowers U^4 norm* and it follows by a theorem of Tao and Ziegler [16] that $\text{rank}(S_4) \leqslant r(\mathbb{F})$ for some constant r.

In the above definitions of rank of a polynomial, we have allowed the function Γ to be arbitrary. It is interesting to ask whether a polynomial is structured in a stronger sense.

▶ **Definition 15** (Strong rank of a polynomial). Given a classical polynomial $P : \mathbb{F}^n \to \mathbb{F}$ of degree d. The *strong rank* of P, denoted by $\text{strong-rank}_d(P)$, is the smallest $r \geqslant 0$, such that there exist nonconstant classical polynomials $G_1, ..., G_r, H_1, ..., H_r : \mathbb{F}^n \to \mathbb{F}$ and a classical polynomial Q such that

- $P(x) = \sum_{i=1}^r G_i H_i + Q$.
- For all $i \in [r]$, we have that $\deg(G_i) + \deg(H_i) \leqslant d$.
- $\deg(Q) \leqslant d - 1$.

The *strong-rank* of a classical polynomial $P : \mathbb{F}^n \to \mathbb{F}$ is equal to $\text{strong-rank}_{\deg(P)}(P)$.

The above notion of rank is somewhat a stronger notion, in particular the following holds for any classical polynomial $P : \mathbb{F}^n \to \mathbb{F}$,

$$\text{rank}(P) \leqslant \text{crank}(P) \leqslant 2 \cdot \text{strong-rank}(P) + 1. \tag{3}$$

Due to the lack of multiplicative structure in $\frac{1}{p^k}\mathbb{Z}/\mathbb{Z}$ for $k > 1$, it is not clear how to define a similar structural notion to strong rank for nonclassical polynomials. Next, we will formalize the notion of a generic collection of polynomials. Intuitively, it should mean that there are no unexpected algebraic dependencies among the polynomials. First, we need to set up some notation.

▶ **Definition 16** (Factors). If X is a finite set then by a *factor* $\mathcal{B}$ we simply mean a partition of X into finitely many pieces called *atoms*.

A finite collection of functions $\phi_1, \dots, \phi_C$ from X to some other space Y naturally define a factor $\mathcal{B} = \mathcal{B}_{\phi_1,\dots,\phi_C}$ whose atoms are sets of the form $\{x : (\phi_1(x), \dots, \phi_C(x)) = (y_1, \dots, y_C)\}$ for some $(y_1, \dots, y_C) \in Y^C$. By an abuse of notation we also use $\mathcal{B}$ to denote the map $x \mapsto (\phi_1(x), \dots, \phi_C(x))$, thus also identifying the atom containing x with $(\phi_1(x), \dots, \phi_C(x))$.

▶ **Definition 17** (Polynomial factors). If $P_1, \dots, P_C : \mathbb{F}^n \to \mathbb{T}$ is a sequence of polynomials, then the factor $\mathcal{B}_{P_1,\dots,P_C}$ is called a *polynomial factor*.

The *complexity* of $\mathcal{B}$, denoted $|\mathcal{B}| := C$, is the number of defining polynomials. The *degree* of $\mathcal{B}$ is the maximum degree among its defining polynomials $P_1, \dots, P_C$. If $P_1, \dots, P_C$ are of depths $k_1, \dots, k_C$, respectively, then the number of atoms of $\mathcal{B}$ is at most $\prod_{i=1}^{C} q^{k_i+1}$ which we denote by $\|\mathcal{B}\|$.

The notions of rank discussed above can now be extended to quantify the structural complexity of a collection of (classical) polynomials.

▶ **Definition 18** (Rank, classical rank, and strong rank of a collection of polynomials). A polynomial factor $\mathcal{B}$ defined by polynomials $P_1, \dots, P_C : \mathbb{F}^n \to \mathbb{T}$ with respective depths $k_1, \dots, k_C$ is said to have rank r if r is the least integer for which there exists $(\lambda_1, \dots, \lambda_C) \in \mathbb{Z}^C$, with $(\lambda_1 \mod q^{k_1+1}, \dots, \lambda_C \mod q^{k_C+1}) \neq 0^C$, such that $\mathrm{rank}_d(\sum_{i=1}^{C} \lambda_i P_i) \leqslant r$, where $d = \max_i \deg(\lambda_i P_i)$.

Given a collection of polynomials $\mathcal{P}$ and a function $r : \mathbb{N} \to \mathbb{N}$, we say that $\mathcal{P}$ is r-regular if $\mathcal{P}$ is of rank larger than $r(|\mathcal{P}|)$. We extend Definition 14 and Definition 15 to classical polynomial factors in a similar manner.

Notice that by the definition of rank, for a degree-d polynomial P of depth k we have

$$\mathrm{rank}(\{P\}) = \min\left\{\mathrm{rank}_d(P), \mathrm{rank}_{d-(q-1)}(qP), \dots, \mathrm{rank}_{d-k(q-1)}(q^k P)\right\},$$

where $\{P\}$ is a polynomial factor consisting of one polynomial P.

In Section 3 we will see that regular collections of polynomials indeed do behave like a generic collection of polynomials in several manners.

Green and Tao [7] and Kaufman and Lovett [11] proved the following relation between bias and rank of a polynomial.

▶ **Theorem 19** ($d < |\mathbb{F}|$ [7], arbitrary $\mathbb{F}$ [11]). *For any $\varepsilon > 0$ and integer $d \geqslant 1$, there exists $r = r(d, \varepsilon, |\mathbb{F}|)$ such that the following is true. If $P : \mathbb{F}^n \to \mathbb{T}$ is a degree-d polynomial* $\mathrm{bias}(P) \geqslant \varepsilon$ *then crank$(P) \leqslant r$.*

More importantly, there are $y_1, \dots, y_r \in \mathbb{F}^n$, and a function $\Gamma : \mathbb{F}^r \to \mathbb{F}$, such that

$$P = \Gamma(D_{y_1}P, \dots, D_{y_r}P).$$

Kaufman and Lovett originally proved Theorem 19 for classical polynomials and classical rank. However, their proof extends to nonclassical polynomials without modification. Note that $r(d, \varepsilon, |\mathbb{F}|)$ does not depend on the dimension n. Motivated by Theorem 19 we define unbiasedness for polynomial factors.

▶ **Definition 20** (Unbiased collection of polynomials). Let $\varepsilon : \mathbb{N} \to \mathbb{R}^+$ be a decreasing function. A polynomial factor $\mathcal{B}$ defined by polynomials $P_1, \dots, P_C : \mathbb{F}^n \to \mathbb{T}$ with respective depths $k_1, \dots, k_C$ is said to be ε-unbiased if for every collection $(\lambda_1, \dots, \lambda_C) \in \mathbb{Z}^C$, with $(\lambda_1 \mod p^{k_1+1}, \dots, \lambda_C \mod p^{k^C+1}) \neq 0^C$ it holds that

$$\left|\mathop{\mathbf{E}}_{x}\left[\mathsf{e}\left(\sum_i \lambda_i P_i(x)\right)\right]\right| < \varepsilon(|\mathcal{B}|).$$

2.3 Regularization of Polynomials

Due to the generic properties of regular factors, it is often useful to *refine* a collection of polynomials to a regular collection [16]. We will first formally define what we mean by refining a collection of polynomials.

▶ **Definition 21** (Refinement). A collection $\mathcal{P}'$ of polynomials is called a *refinement* of $\mathcal{P} = \{P_1, ..., P_m\}$, and denoted $\mathcal{B}' \succeq \mathcal{B}$, if the induced partition by $\mathcal{B}'$ is a combinatorial refinement of the partition induced by $\mathcal{B}$. In other words, if for every $x, y \in \mathbb{F}^n$, $\mathcal{B}'(x) = \mathcal{B}'(y)$ implies $\mathcal{B}(x) = \mathcal{B}(y)$.

Green and Tao [7], showed that given any nondecreasing function $r : \mathbb{N} \to \mathbb{N}$, any classical polynomial factor can be refined to an r classical-rank classical factor. The basic idea is simple; if some classical polynomial has low rank, decompose it to a few lower degree classical polynomials, and repeat. The formal proof uses a transfinite induction on the number of classical polynomials of each degree which defines the classical polynomial factor. The bounds on the number of classical polynomials obtained in the regularization process have Ackermann-type dependence on the degree d, even when the regularity parameter $r(\cdot)$ is a “reasonable" function. As such, it gives nontrivial results only for constant degrees. The extension of this regularity lemma to nonclassical polynomials is more involved, and was proved by Tao and Ziegler [16] as part of their proof of the inverse Gowers theorem.

▶ **Theorem 22** (Regularity lemma for (nonclassical) polynomials [16]). *Let $r : \mathbb{N} \to \mathbb{N}$ be a non-decreasing function and $d \geqslant 1$ be an integer. Then, there is a function $C_{\mathbb{F},r,d} : \mathbb{N} \to \mathbb{N}$ such that the following holds. Suppose $\mathcal{B}$ is a factor defined by polynomials $P_1, \ldots, P_C : \mathbb{F}^n \to \mathbb{T}$ of degree at most d. Then, there is an r-regular factor $\mathcal{B}'$ consisting of polynomials $Q_1, \ldots, Q_{C'} : \mathbb{F}^n \to \mathbb{T}$ of degree $\leqslant d$ such that $\mathcal{B}' \succeq \mathcal{B}$ and $C' \leqslant C_{22}^{(\mathbb{F},r,d)}(C)$.*

3 Properties of rank, crank, and strong-rank

A high-rank polynomial of degree d is, intuitively, a “generic” degree d polynomial; there are no unexpected ways to decompose it into lower degree polynomials. In this section we make precise this intuition.

Using a standard observation that relates the bias of a function to its distribution on its range, Theorem 19 implies that high-rank polynomials behave like independent random variables. See [1, 10] for further discussion of stronger equidistribution properties of high-rank polynomials.

Another way that high-rank polynomials behave like generic polynomials is that their restriction to subspaces preserves degree and high rank. We refer to [1, 3] for a proof.

▶ **Lemma 23** (Degree and rank preservation). *Suppose $f : \mathbb{F}^n \to \mathbb{T}$ is a polynomial of degree d and rank $\geqslant r$, where $r > q + 1$. Let A be a hyperplane in $\mathbb{F}^n$. Then, $f|_A$ is a polynomial of degree d and rank $\geqslant \max\{r - d - 1, r - |\mathbb{F}| - 1\}$, unless $d = 1$ and f is constant on A.*

The following is a surprising and very useful property of high-rank polynomials that was proved by Bhattacharyya, et. al. [1].

▶ **Lemma 24** (Degree preservation, Lemma 2.13 of [1]). *Let $d > 0$ be given. There exists a nondecreasing function $r_{d,\mathbb{F}} : \mathbb{N} \to \mathbb{N}$ such that the following holds. Let $\mathcal{B}$ be a rank $\geqslant r_{d,\mathbb{F}}$ polynomial factor defined by degree $\leqslant d$ polynomials $P_1, ..., P_m : \mathbb{F}^n \to \mathbb{T}$. Let $\Gamma : \mathbb{T}^n \to \mathbb{T}$. Then*

$$\deg(\Gamma(Q_1(x), ..., Q_m(x))) \leqslant \deg(\Gamma(P_1(x), ..., P_m(x))),$$

for every collection of polynomial $Q_1, ..., Q_m : \mathbb{F}^n \to \mathbb{T}$*, with* $\deg(Q_i) \leqslant \deg(P_i)$ *and* $\operatorname{depth}(Q_i) \leqslant \operatorname{depth}(P_i)$*.*

We prove a lemma relating the strong-rank of a polynomial to its strong-rank over constant codimensional affine subspaces.

▶ **Lemma 25.** *Let* $f : \mathbb{F}^n \to \mathbb{F}$ *be a degree* d *classical polynomial and* V *be an affine subspace of* $\mathbb{F}^n$ *of dimension* $n - t$*. Then,*

$$\textit{strong-rank}(f) \leqslant \textit{strong-rank}(f|_V) + t.$$

Proof. It suffices to prove that for a hyperplane W, strong-rank$(f) \leqslant$ strong-rank$(f|_V) + 1$. The lemma then simply follows by induction on t, the codimension of V.

Suppose $W = \{x \in \mathbb{F}^n | \sum_{i=1}^n w_i x_i = a\}$, where $w \in \mathbb{F}^n$ and $a \in \mathbb{F}$. Applying an affine invertible projection, we can assume without loss of generality that $w = (1, 0, \ldots, 0)$ and $a = 0$, and thus $W = \{x \in \mathbb{F}^n | x_1 = 0\}$. Assume that strong-rank$(f|_W) = r$, hence there exist nonconstant classical polynomials $G_1, ..., G_r, H_1, ..., H_r : W \to \mathbb{F}$ where $\deg(G_i)+\deg(H_i) \leqslant d$ and a degree $\leqslant d - 1$ classical polynomial $Q : W \to \mathbb{F}$ such that

$$f|_W = \sum_{i=1}^{r} G_i H_i + Q.$$

Now note that,

$$f(x_1, ..., x_n) = f|_W(0, x_2, \ldots, x_n) + x_1 R(x_1, ..., x_n),$$

where $\deg(R) \leqslant d - 1$. Thus

$$f = x_1 R + \sum_{i=1}^{r} G_i H_i + Q,$$

equivalently strong-rank$(f) \leqslant r + 1$. ◀

The above lemma shows that high strong-rank classical polynomials are generic in a strong sense. We finally observe that all the discussed notions of rank are subadditive.

▶ **Claim 26.** *For every fixed vectors* $a, b \in \mathbb{F}^n$*,*
(i) *strong-rank*$(D_{a+b} f) \leqslant$ *strong-rank*$(D_a f) +$ *strong-rank*$(D_b f)$*.*
(ii) *crank*$(D_{a+b} f) \leqslant$ *crank*$(D_a f) +$ *crank*$(D_b f)$*.*
(iii) *rank*$(D_{a+b} f) \leqslant$ *rank*$(D_a f) +$ *rank*$(D_b f)$*.*

Proof. We compute $D_{a+b} f(x)$,

$$D_{a+b} f(x) = D_b f(x + a) + D_a f(x).$$

The claim follows since $\tau(D_b f(x+a)) \leqslant \tau(D_b f(x))$ for any choice of $\tau \in \{\text{strong-rank}, \text{crank}, \text{rank}\}$, as the degrees of polynomials are preserved under affine shifts. ◀

4 Structure of biased polynomials

Throughout the paper we will assume $\mathbb{F} = \mathbb{F}_q$ is a fixed prime field.

We will need the following theorem of Sanders [15] on the structure of sets with small doubling.

▶ **Theorem 27** ([15]). *Suppose $A \subseteq \mathbb{F}^n$ satisfies $\frac{|A|}{|G|} \geqslant \alpha$. Then $A + A + A = \{a_1 + a_2 + a_3 | a_1, a_2, a_3 \in A\}$ contains an affine subspace of codimension at most $O_{|\mathbb{F}|,\alpha}(1)$.*

The following lemma states that for a function $f : \mathbb{F}^n \to \mathbb{F}$ to be biased, there must be a positive set of directions y for which $D_y f$ is somewhat biased.

▶ **Lemma 28.** *Suppose $f : \mathbb{F}^n \to \mathbb{F}$ is such that $\mathrm{bias}(f) = \delta$. Then there exists a set $A \subseteq \mathbb{F}^n$, with $|A| \geqslant \frac{\delta^2}{2}|\mathbb{F}|^n$ such that for every $y \in A$, $\mathrm{bias}(D_y f) \geqslant \frac{\delta^2}{2}$.*

Proof. We compute the average bias of $D_y f$ for $y \in \mathbb{F}^n$ uniformly at random.

$$\mathop{\mathbf{E}}_{y\in\mathbb{F}^n}[\mathrm{bias}(D_y f)] = \mathop{\mathbf{E}}_{y\in\mathbb{F}^n}\left[|\mathop{\mathbf{E}}_{x\in\mathbb{F}^n} e_{\mathbb{F}}(f(x+y) - f(x))|\right] \geqslant |\mathop{\mathbf{E}}_{z,x\in\mathbb{F}^n}[e_{\mathbb{F}}(f(z))e_{\mathbb{F}}(-f(x))]| = \delta^2. \tag{4}$$

Thus, since $\mathrm{bias}(f) \leqslant 1$, we get

$$\mathop{\mathbf{Pr}}_{y\in\mathbb{F}^n}\left[\mathrm{bias}(D_y f) \geqslant \frac{\delta^2}{2}\right] \geqslant \frac{\delta^2}{2}. \tag{5}$$

The lemma follows by choosing $A := \{y \in \mathbb{F}^n | \mathrm{bias}(D_y f) \geqslant \frac{\delta^2}{2}\} \subseteq \mathbb{F}^n$. ◀

We will use this lemma along with Theorem 27 and Claim 26 to show that for every biased function f there exists a not too small subspace restricted to which all the derivatives of f are biased.

4.1 Structure of biased polynomials I, when $d < |\mathbb{F}| + 4$

In this section we prove that biased degree d classical polynomials are strongly structured when $d < |\mathbb{F}| + 4$.

▶ **Theorem 7** (restated – Biased degree d polynomials I (when $d < |\mathbb{F}|+4$)). *Suppose $d > 0$ and $\mathbb{F}$ be a prime field satisfying $d < |\mathbb{F}|+4$. Let $f : \mathbb{F}^n \to \mathbb{F}$ be a degree d classical polynomial with $\mathrm{bias}(f) = \delta$. Then strong-rank$(f) \leqslant c(\delta, d, q)$, namely there exists $c_7 \leqslant c(\delta, d, q)$, nonconstant classical polynomials $G_1, ..., G_c, H_1, ..., H_c : \mathbb{F}^n \to \mathbb{F}$ and a classical polynomial $Q : \mathbb{F}^n \to \mathbb{F}$ such that the following hold.*
- $f = \sum_{i=1}^{c} G_i H_i + Q$.
- *For every $i \in [c]$, $\deg(G_i) + \deg(H_i) \leqslant d$.*
- $\deg(Q) \leqslant d - 1$.

Note that c_7 does not depend on n.

Proof. By Lemma 28 there exists a set $A \subseteq \mathbb{F}^n$, with $|A| \geqslant \frac{\delta^2}{2}|\mathbb{F}|^n$ such that for every $y \in A$,

$$\mathrm{bias}(D_y f) \geqslant \frac{\delta^2}{2}.$$

Thus by Theorem 19 for every $y \in A$,

$$\mathrm{crank}(D_y f) \leqslant r = r_{19}(d, |\mathbb{F}|, \delta).$$

Applying Theorem 27, there is a subspace $V \subset \mathbb{F}^n$ of co-dimension $t = O_{\delta,|\mathbb{F}|}(1)$ and $h_0 \in \mathbb{F}^n$ such that $V + h_0 \subseteq A + A + A$. By Claim 26 (ii), since $V + h_0 \subseteq A + A + A$ we have that for every $y \in V$,

$$\mathrm{crank}(D_y f) \leqslant c_1 \leqslant 3r.$$

By a simple averaging argument, there is an affine shift of V, $W := V + h$ such that $\text{bias}(f|_W) \geqslant \delta$. Let us denote $\widetilde{f} := f|_W$. By Lemma 25, it is sufficient to prove that $\text{strong-rank}(\widetilde{f}) \leqslant c_1(|\mathbb{F}|, \delta)$. Since $\text{bias}(\widetilde{f}) \geqslant \delta$, Theorem 19 implies $\text{crank}(\widetilde{f}) \leqslant r_0 = r_0(\delta, |\mathbb{F}|)$. Moreover, there are $y_1, \dots, y_{r_0} \in W$ and a $\Gamma : \mathbb{F}^{r_0} \to \mathbb{F}$ such that

$$\widetilde{f} = \Gamma(D_{y_1}\widetilde{f}, \dots, D_{y_{r_0}}\widetilde{f}). \tag{6}$$

Note that for all $i \in [r_0]$,

$$\text{crank}_{d-1}(D_{y_i}\widetilde{f}) \leqslant \text{crank}(D_{y_i} f) \leqslant c_0 \tag{7}$$

This is due to the fact that an affine transformation can only decrease the degrees of classical polynomials and thus it can only decrease the crank of classical polynomials.

▶ Remark. We point out that the subscript $d-1$ in the LHS of (7) is necessary, as can be seen by the following example. Suppose $d-1 = 4$, $m > 0$ and $n = 3m+4$. Let $Q = x_{n-3}x_{n-2}x_{n-1}x_n + \sum_{i=1}^m x_{3i-2}x_{3i-1}x_{3i}$. Now note that

$$\text{crank}(Q) \leqslant 3,$$

while

- $\text{crank}(Q|_{x_n=0}) = \text{crank}(\sum_{i=1}^m x_{3i-2}x_{3i-1}x_{3i}) = \omega_n(1)$, since $\|e_{\mathbb{F}}(\sum_{i=1}^m x_{3i-2}x_{3i-1}x_{3i})\|_{U^3} = o(1)$.
- $\text{crank}_4(Q|_{x_n=0}) = 1$, since $\deg(Q|_{x_n=0}) < 4$.

By (7) there exist degree $\leqslant d-2$ classical polynomials $\left\{G_1^{(i)}, \dots, G_{c_0}^{(i)}\right\}_{i=1}^{r_0}$ and a function $\Lambda : \mathbb{F}^{r_0 c_0} \to \mathbb{F}$ such that

$$\widetilde{f} = \Lambda\left((G_1^{(i)}, \dots, G_{c_0}^{(i)})_{i=1}^{r_0}\right). \tag{8}$$

We would like to regularize this collection of classical polynomials, however we would like to avoid any appearance of nonclassical polynomials. The following observation allows us to do exactly that as long as $d < |\mathbb{F}| + 4$.

▶ **Claim 29** (Nonclassical regularity lemma over large characteristic). *Let $r : \mathbb{N} \to \mathbb{N}$ be a non-decreasing function. And d be such that $d < |\mathbb{F}| + 4$. Then, there is a function $C_{29}^{\mathbb{F},r} : \mathbb{N} \to \mathbb{N}$ such that the following holds. Suppose $\mathcal{B}$ is a factor defined by classical polynomials $P_1, \dots, P_C : \mathbb{F}^n \to \mathbb{T}$ of degree at most $d-2$. Then, there is an r-regular factor $\mathcal{B}'$ consisting only of classical polynomials $Q_1, \dots, Q_{C'} : \mathbb{F}^n \to \mathbb{T}$ of degree $\leqslant d-2$ such that $\mathcal{B}' \succeq_{sem} \mathcal{B}$ and $C' \leqslant C_{29}^{(\mathbb{F},r)}(C)$.*

▶ Remark. Note that the above claim does not hold for general degrees, as we require the obtained factor be high-rank as defined in Definition 13, which is complexity against (nonclassical) polynomials. To see this, we observe that in the case of quartic classical polynomials, the single classical polynomial $\{S_4\}$ cannot be refined to a high-rank polynomial factor defined by $O(1)$ *classical* polynomials. However, it can be refined to a high-rank nonclassical factor by Theorem 22. This is the barrier to extending our results to sextic and higher-degree classical polynomials. Starting with a biased sextic classical polynomial, dealing with non-classical polynomials seems to be unavoidable.

We postpone the proof of Claim 29 and show how it can be used to conclude Theorem 4. Fix $r_1 : \mathbb{N} \to \mathbb{N}$ a nondecreasing function as in Lemma 24 for degree $d-2$. Let $\mathcal{B}$ be the factor defined by degree $\leqslant d-2$ classical polynomials $\{G_1^{(i)}, \dots, G_{c_0}^{(i)}\}_{i=1}^{r_0}$. Applying Claim 29 to $\mathcal{B}$ with regularity parameter r_1, we obtain a refinement $\mathcal{B}' \succeq_{sem} \mathcal{B}$, where $\mathcal{B}'$ is defined by

$c_2 := C_{29}^{(\mathbb{F},r_1)}(c_0r_0)$ classical degree $\leqslant d-2$ polynomials $R_1,\dots,R_{c_2} : \mathbb{F}^n \to \mathbb{F}$. Namely, there exists a function $\mathcal{K} : \mathbb{F}^{c_2} \to \mathbb{F}$, such that

$$\widetilde{f} = \mathcal{K}(R_1,\dots,R_{c_2}).$$

Applying an affine transformation, assume without loss of generality that $W = \{x \in \mathbb{F}^n | x_1 = x_2 = \cdots = x_t = 0\}$. Moreover, we may assume that $n-t > c_2$, since otherwise, $\widetilde{f}$ has at most $d(n-t)^d = O(c_2^d)$ monomials, making the theorem statement trivial. For every $i \in [c_2]$, let $d_i := \deg(R_i)$, $s_i := \sum_{j=1}^{i} d_i$, and define $R'_i := x_{s_{i-1}+1}\cdots x_{s_i}$. We have that $\deg(R'_i) = \deg(R_i)$ and thus by Lemma 24,

$$\deg(\mathcal{K}(R'_1,\dots,R'_{c_2})) \leqslant \deg(\mathcal{K}(R_1,\dots,R_{c_2})) = \deg(\widetilde{f}) = d.$$

Note that $\mathcal{K} : \mathbb{F}^{c_2} \to \mathbb{F}$ is a classical polynomial, and R'_1, ..., R'_{c_2} are monomials on disjoint variables, thus plugging in R'_is into $\mathcal{K}$'s variables, no cancelations can occur. In particular,

$$\mathcal{K}(y_1,\dots,y_{c_2}) = \sum_{s\in\{0,\dots,q-1\}^{c_2},\sum_i s_id_i\leqslant d} \alpha_s \prod_{i\in S} y_i^{s_i},$$

where $\alpha_S \in \mathbb{F}$ are coefficients of $\mathcal{K}$. Hence,

$$\widetilde{f} = \mathcal{K}(R_1,\dots,R_{c_2}) = \sum_{s\in\{0,\dots,q-1\}^{c_2},\sum_i s_id_i\leqslant d} \alpha_s \prod_{i\in S} R_i^{s_i}. \tag{9}$$

Namely, strong-rank$(\widetilde{f}) \leqslant dc_2^d$, and by Lemma 25 we deduce strong-rank$(f) \leqslant dc_2^d + t$ as desired. ◀

Proof of Claim 29. We observe that the iterative proof of Theorem 22 can be modified to include only classical polynomials. Theorem 22 is proved by a transfinite induction on the vector of number of (possibly nonclassical) polynomials of each degree and depth defining the polynomial factor. One then argues that a polynomial factor that is not of the desired rank, can always be refined to a polynomial factor where some polynomial is replaced by a collection of polynomials that are of either lower degree, or same degree with lower depth.

We now make use of the fact that $d < |\mathbb{F}| + 4$. We observe that if we start with a polynomial factor defined by degree $\leqslant d-2$ classical polynomials, the only nonclassical polynomials that may arise are of degree $d-3 \leqslant |\mathbb{F}|$ and thus of depth 1, this is due to the fact that any nonclassical polynomial of depth $\geqslant 2$ has degree $\geqslant 2|\mathbb{F}|-1$. Now we use a known fact that polynomials of degree $|\mathbb{F}|$ that are not classical are unncessary in higher order Fourier analysis. More precisely in the inverse theorem for Gowers norms of [16] for the case of degree $|\mathbb{F}|$ polynomials, one can assume that the polynomial $P : \mathbb{F}^n \to \mathbb{T}$ in the statement of the theorem is a *classical* polynomial of degree at most $\leqslant |\mathbb{F}|$. More generally [9] showed a similar fact for higher depths.

▶ **Theorem 30** (Unnecessary depths [9]). *Let $k \geqslant 1$, and q the characteristic of $\mathbb{F}$. Every nonclassical polynomial $f : \mathbb{F}^n \to \mathbb{T}$ of degree $1+k(q-1)$ and depth k, can be expressed as a function of three degree $\leqslant 1+k(q-1)$ polynomials of depth $\leqslant k-1$.*

By the above discussion we may assume that in our application of Theorem 22, $\mathcal{B}'$ is defined via only classical polynomials. ◀

4.2 Structure of biased polynomials II, when $d < |\mathbb{F}| + 4$

In this section we prove that a biased degree d classical polynomial is constant on a large subspace.

▶ **Theorem 8** (restated – Biased degree d polynomials II (when $d < |\mathbb{F}| + 4$)). *Suppose $d > 0$ and $\mathbb{F}$ be a prime field satisfying $d < |\mathbb{F}|+4$. Let $f : \mathbb{F}^n \to \mathbb{F}$ be a degree d classical polynomial with* $\text{bias}(f) = \delta$. *There exists an affine subspace V of dimension $\Omega_{d,\delta}(n^{1/\lfloor \frac{d-2}{2} \rfloor})$ such that $f|_V$ is a constant.*

In the case of $d = 5$ we have $5 < 2 + 4 \leqslant |\mathbb{F}| + 4$ and $\lfloor (d-2)/2 \rfloor = 1$, hence we obtain a subspace of dimension $\Omega_\delta(n)$ as desired in Theorem 5.

We will need the following result of Cohen and Tal [5] on the structure of low degree polynomials.

▶ **Theorem 31** ([5], Theorem 3.5). *Let q be a prime power. Let $f_1, \dots, f_\ell : \mathbb{F}_q^n \to \mathbb{F}_q$ be (classical) polynomials of degree $d_1, \dots, d_\ell$ respectively. Let k be the least integer such that*

$$n \leqslant k + \sum_{j=0}^{\ell} (d_i + 1) \sum_{j=0}^{d_i - 1} (d_i - j) \cdot \binom{k+j-1}{j}.$$

Then, for every $u_0 \in \mathbb{F}_q^n$ there exists a subspace $U \subseteq \mathbb{F}_q^n$ of dimension k, such that for all $i \in [\ell]$, f_i restricted to $u_0 + U$ is a constant function.

In particular, if $d_1, ..., d_\ell \leqslant d$, then the above holds for $k = \Omega((n/\ell)^{\frac{1}{d-1}})$.

Proof of Theorem 8. Following the proof of Theorem 7, there exists an affine subspace W of dimension $n - t$ for $t = poly(\log(\frac{1}{\delta^2}))$, for which (9) holds. By Theorem 19, choosing a proper regularity parameter in the application of Claim 29, we can further assume that the factor defined by $R_1, ..., R_{c_2}$ is $\frac{\delta}{2} q^{-c_2}$-unbiased in the sense of Definition 20. We may rewrite (9) in the form

$$f|_W = \sum_{i=1}^{C} \alpha_i G_i H_i + M,$$

where $C \leqslant c_2^d$, α_i are field elements, M is a degree $\leqslant d-2$ classical polynomial, G_is and H_is are nonconstant degree $\leqslant d-2$ classical polynomials satisfying $\deg(G_i) + \deg(H_i) \leqslant d$. Moreover, every G_i and H_i is product of a subset of $\{R_1, ..., R_{c_2}\}$. We crucially observe that M can be taken to be of the form

$$M = \sigma_0 + \sum_{i=1}^{c_2} \sigma_i R_i,$$

where σ_i are field elements, such that $\sigma_i \neq 0$ implies that R_i does not appear in $\sum_{i=1}^{C} \alpha_i G_i H_i$.

▶ **Claim 32.** *Let f, W, $R_1, ...R_{c_2}$ and M be as above. Then M is a constant.*

Proof. Assume for contradiction that M is nonconstant. By the above discussion, letting

$$S := \{j \in [c_2] : R_j \text{ appears in } \sum_i \alpha_i G_i H_i\},$$

we have

$$f|_W = \Lambda(R_j)_{j \in S} + \sum_{i \in [c_2] \setminus S} \sigma_j R_j,$$

for some function $\Lambda : \mathbb{F}^{|S|} \to \mathbb{F}$. Writing the Fourier expansion of $e_{\mathbb{F}}(\Lambda)$, we have

$$e_{\mathbb{F}}(f|_W) = \sum_{\gamma \in \mathbb{F}^{|\S|}} \widehat{\Lambda}(\gamma) e_{\mathbb{F}}(\sum_{j\in S} \gamma_j R_j + M).$$

Note that W was chosen such that $\mathrm{bias}(f|_W) \geqslant \delta$. Thus,

$$\begin{aligned}\mathrm{bias}(f|_W) &= |\mathop{\mathbf{E}}_{x\in\mathbb{F}^n} e_{\mathbb{F}}(\Lambda(R_j)_{j\in S} + M)| \\ &= |\mathop{\mathbf{E}}_{x\in\mathbb{F}^n} \sum_{\gamma\in\mathbb{F}^{|S|}} \widehat{\Lambda}(\gamma) e_{\mathbb{F}}(M + \sum_{j\in S}\gamma_j R_j)| \\ &\leqslant \sum_{\gamma\in\mathbb{F}^{|S|}} |\widehat{\Lambda}(\gamma)| \cdot \mathrm{bias}(M + \sum_{j\in S}\gamma_j R_j) \\ &\leqslant q^{c_2} \cdot \frac{\delta}{2} q^{-c_2} < \delta,\end{aligned}$$

contradicting $\mathrm{bias}(f|_W) = \delta$, where the last inequality uses the fact that the factor defined by $R_1, ..., R_{c_2}$ is $\frac{\delta}{2}q^{-c_2}$-unbiased. ◀

By the above claim M is a constant, and thus

$$f|_W = \sigma_0 + \sum_{i=1}^{C} \alpha_i G_i H_i.$$

Recall that $\deg(G_i) + \deg(H_i) \leqslant d$, hence for every i, $\min\{\deg(G_i), \deg(H_i)\} \leqslant \lfloor \frac{d}{2} \rfloor$. Thus by Theorem 31, there is an $\Omega_C((n-t)^{1/\lfloor \frac{d-2}{2} \rfloor}) = \Omega_{\delta,\mathbb{F},d}(n^{1/\lfloor \frac{d-2}{2}\rfloor})$ dimensional affine subspace W' such that $f|_{W'}$ is constant. ◀

5 Algorithmic Aspects

In this section we show that the strong structures implied by Theorem 4 and Theorem 7 can be found by a deterministic algorithm that runs in time polynomial in n.

▶ **Theorem 9** (restated). *Suppose $\delta > 0$, $d > 0$ are given, and let $\mathbb{F}$ be a prime field satisfying $d < |\mathbb{F}| + 4$. There is a deterministic algorithm that runs in time $O(n^{O(d)})$ and given as input a degree d classical polynomial $f : \mathbb{F}^n \to \mathbb{F}$ satisfying* $\mathrm{bias}(f) = \delta$, *outputs a number $c \leqslant c(\delta, |\mathbb{F}|, d)$, a collection of degree $\leqslant d-1$ classical polynomials $G_1, ..., G_c, H_1, ..., H_c : \mathbb{F}^n \to \mathbb{F}$ and a classical polynomial $Q : \mathbb{F}^n \to \mathbb{F}$, such that*
- $f = \sum_{i=1}^{c} G_i H_i + Q$.
- *For every $i \in [c]$,* $\deg(G_i) + \deg(H_i) \leqslant d$.
- $\deg(Q) \leqslant d - 1$.

Proof. We will use the following result of Bhattacharyya, et. al. [2] who proved several algorithmic regularity lemmas for polynomials.

▶ **Theorem 33** ([2], Theorem 1.6). *For every finite field $\mathcal{F}$ of fixed prime order, positive integers d, k, every vector of positive integers $\Delta = (\Delta_1, ..., \Delta_k)$, and every function $\Gamma : \mathbb{F}^k \to \mathbb{F}$, there is a deterministic algorithm that takes as input a classical polynomial $f : \mathbb{F}^n \to \mathbb{F}$ of degree d, runs in time polynomial in n, and outputs classical polynomials $Q_1, ..., Q_k$ of degrees respectively at most $\Delta_1, ..., \Delta_k$ such that*

$$f = \Gamma(Q_1, ..., Q_k),$$

if such a decomposition exists, while otherwise accurately returning NO.

By Theorem 7, we know that there is $c \leqslant C(\delta, |\mathbb{F}|, d)$ such that there exist a collection of nonconstant classical polynomials $G_1, ..., G_c, H_1, ..., H_c : \mathbb{F}^n \to \mathbb{F}$, and a classical polynomial $Q : \mathbb{F}^n \to \mathbb{F}$, such that

$$f = \sum_{i=1}^{c} G_i H_i + Q, \tag{10}$$

for every $i \in [c]$, $\deg(G_i) + \deg(H_i) \leqslant d$, and $\deg(Q) \leqslant d-1$. The algorithm is now straight-forward.

```
**1** Iterate through all choices for $c \leqslant C(\delta, |\mathbb{F}|, d)$. This is our guess for the number of terms
  in the summation in (10).
  **1.1** Iterate through all choices of $d_1, \ldots, d_c, d'_1, \ldots, d'_c \leqslant d-1$ and $d'' \leqslant d-1$ such that
    $d_i + d'_i \leqslant d$. These are our guesses for degree sequences for $G_1, ..., G_c, H_1, ..., H_c$ and
    $Q$. Note that this step does not depend on $n$.
    **1.1.1** Define $\Gamma : \mathbb{F}^{2c+1} \to \mathbb{F}$ as
        $$\Gamma(x_1, \ldots, x_c, y_1, \ldots, y_c, z) := \sum_{i=1}^{c} x_i y_i + z.$$
    **1.1.2** Run Theorem 33 on the classical polynomial $f$, with $\Delta = (d_1, \ldots, d_c, d'_1, \ldots, d'_c,$
      $d'')$ and $\Gamma$ as inputs.
      **1.1.2.a** If the algorithm outputs NO, then continue.
      **1.1.2.b** If the algorithm outputs a collection of classical polynomials satisfying the
        decomposition, halt and output the desired decomposition.
```

By Theorem 7 and Theorem 33 the above algorithm will always halt with a decomposition of desired form. The number of possible choices in **1** and **1.1** do not depend on n, and step **1.1.2** runs in polynomial time in n, as a result making the algorithm polynomial time in n. ◀

6 Conclusions

Green and Tao [7] and Kaufman and Lovett [11] proved that every degree d classical polynomial f with $\text{bias}(f) = \delta$ can be written in the form

$$f = \Gamma(P_1, ..., P_c), \tag{11}$$

for $c \leqslant c(\delta, d, \mathbb{F})$ and degree $\leqslant d-1$ classical polynomials $P_1, ..., P_c$. However, nothing is known on the structure of the function Γ in (11). In this work we showed that in the case of degree five polynomials we can say more about the structure of f. More generally for degree d classical polynomials when $d < |\mathbb{F}| + 4$, we can write

$$f = \sum_{i=1}^{C} G_i H_i + Q,$$

for nontrivial classical polynomials G_i, H_i satisfying $\deg(G_i) + \deg(H_i) \leqslant d$, and $\deg(Q) \leqslant d-1$. It is a fascinating question whether similar structure theorems hold in the case of $d \geqslant |\mathbb{F}| + 4$, more specifically we suspect that answering this question for degree 6 classical polynomials and $\mathbb{F} = \mathbb{F}_2$ will suffice resolve the question for all degrees and characteristics.

▶ **Open Problem 34.** *Can every biased degree six classical polynomial $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be written in the form*

$$f = \sum_{i=1}^{C} G_i H_i + Q,$$

for $C \leqslant C(\text{bias}(f))$, nontrivial classical polynomials Q, G_i, H_i satisfying $\deg(G_i)+\deg(H_i) \leqslant 6$, and $\deg(Q) \leqslant 5$?

A somewhat weaker question that also remains open is whether we can bound the degree of Γ in (11) in terms of d only.

▶ **Open Problem 35.** *Suppose that $\mathbb{F}$ is a prime field. Can every degree d classical polynomial $f : \mathbb{F}^n \to \mathbb{F}$ be written in the form*

$$f = \Gamma(P_1, ..., P_{C_1}),$$

where $C \leqslant C(\text{bias}(f), \mathbb{F}, d)$, $P_1, ..., P_C$ are degree $\leqslant d-1$ classical polynomials, and $\deg(\Gamma) \leqslant O_d(1)$?

Finally, we note that the constants obtained in Theorems 4, 5, 7 and 8, unlike Theorems 2 and 3, have very bad dependence on δ and d. In particular, in the case of degree five polynomials, an interesting problem that remains unaddressed is to find out what the optimum constant achievable in Theorem 4 is.

Acknowledgements. We thank Avishay Tal and Avi Wigderson for helpful discussions. We also thank the referees for their useful comments.

References

1 Arnab Bhattacharyya, Eldar Fischer, Hamed Hatami, Pooya Hatami, and Shachar Lovett. Every locally characterized affine-invariant property is testable. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC'13, pages 429–436, New York, NY, USA, 2013. ACM. doi:10.1145/2488608.2488662.

2 Arnab Bhattacharyya, Pooya Hatami, and Madhur Tulsiani. Algorithmic regularity for polynomials and applications. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1870–1889, 2015. doi:10.1137/1.9781611973730.125.

3 Abhishek Bhowmick and Shachar Lovett. Bias vs structure of polynomials in large fields, and applications in effective algebraic geometry and coding theory. *CoRR*, abs/1506.02047, 2015. URL: http://arxiv.org/abs/1506.02047.

4 Abhishek Bhowmick and Shachar Lovett. The list decoding radius of reed-muller codes over small fields. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC'15, pages 277–285, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746543.

5 Gil Cohen and Avishay Tal. Two structural results for low degree polynomials and applications. *RANDOM*, 2015.

6 Leonard Eugene Dickson. *Linear groups: With an exposition of the Galois field theory.* with an introduction by W. Magnus. Dover Publications, Inc., New York, 1958.

7 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.

8 Elad Haramaty and Amir Shpilka. On the structure of cubic and quartic polynomials. In *STOC'10 – Proceedings of the 2010 ACM International Symposium on Theory of Computing*, pages 331–340. ACM, New York, 2010.

9 Hamed Hatami, Pooya Hatami, and James Hirst. Limits of Boolean functions on $\mathbb{F}_p^n$. *Electron. J. Combin.*, 21(4):Paper 4.2, 15, 2014.

10 Hamed Hatami, Pooya Hatami, and Shachar Lovett. General systems of linear forms: equidistribution and true complexity. *Advances in Mathematics*, 292:446–477, 2016.

11 Tali Kaufman and Shachar Lovett. Worst case to average case reductions for polynomials. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:166–175, 2008. doi:10.1109/FOCS.2008.17.

12 Tali Kaufman, Shachar Lovett, and Ely Porat. Weight distribution and list-decoding size of reed–muller codes. *Information Theory, IEEE Transactions on*, 58(5):2689–2696, 2012.

13 Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications.* Cambridge university press, 1994.

14 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.

15 Tom Sanders. Additive structures in sumsets. *Math. Proc. Cambridge Philos. Soc.*, 144(2):289–316, 2008. doi:10.1017/S030500410700093X.

16 Terence Tao and Tamar Ziegler. The inverse conjecture for the Gowers norm over finite fields in low characteristic. *Ann. Comb.*, 16(1):121–188, 2012. doi:10.1007/s00026-011-0124-3.

Lower Bounds on Same-Set Inner Product in Correlated Spaces*

Jan Hązła†1, Thomas Holenstein2, and Elchanan Mossel‡3

1 **ETH Zürich, Department of Computer Science, Zurich, Switzerland**
`jan.hazla@inf.ethz.ch`
2 **Google, Zurich, Switzerland**
`thomas.holenstein@gmail.com`
3 **MIT, Cambridge MA, USA**
`elmos@mit.edu`

Abstract

Let $\mathcal{P}$ be a probability distribution over a finite alphabet Ω^ℓ with all ℓ marginals equal. Let $\underline{X}^{(1)}, \ldots, \underline{X}^{(\ell)}, \underline{X}^{(j)} = (X_1^{(j)}, \ldots, X_n^{(j)})$ be random vectors such that for every coordinate $i \in [n]$ the tuples $(X_i^{(1)}, \ldots, X_i^{(\ell)})$ are i.i.d. according to $\mathcal{P}$.

The question we address is: does there exist a function $c_{\mathcal{P}}()$ independent of n such that for every $f : \Omega^n \to [0,1]$ with $\mathrm{E}[f(\underline{X}^{(1)})] = \mu > 0$:

$$\mathrm{E}\left[\prod_{j=1}^{\ell} f(\underline{X}^{(j)})\right] \geq c_{\mathcal{P}}(\mu) > 0\ ?$$

We settle the question for $\ell = 2$ and when $\ell > 2$ and $\mathcal{P}$ has bounded correlation $\rho(\mathcal{P}) < 1$.

1998 ACM Subject Classification G.2.1 Combinatorics, G.3 Probabilistic algorithms

Keywords and phrases same set hitting, product spaces, correlation, lower bounds

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.34

1 Introduction

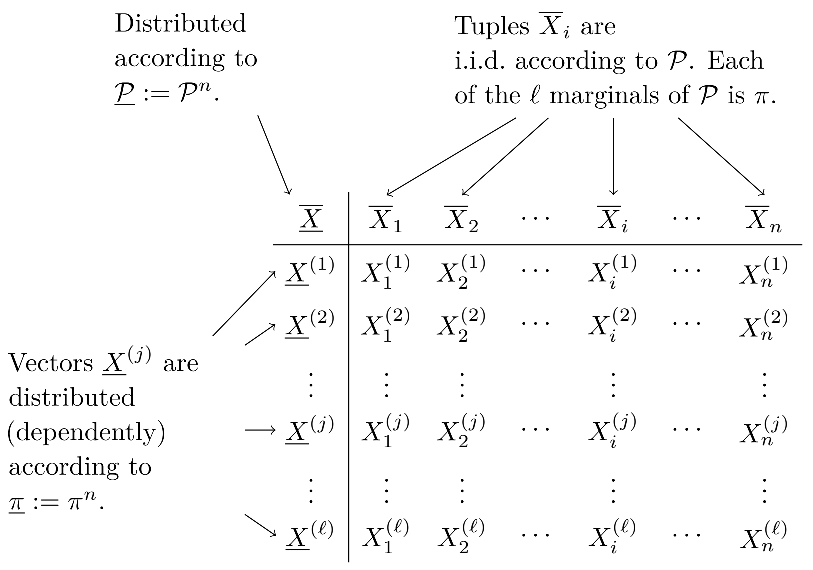
1.1 Basic example

To introduce the problem we are studying, consider the following example. Let $S \subseteq \{0,1,2\}^n$ be a non-empty set of density $\mu = \frac{|S|}{3^n}$. We pick a random vector $\underline{X} = (X_1, \ldots, X_n)$ uniformly from $\{0,1,2\}^n$, and then sample another vector $\underline{Y} = (Y_1, \ldots, Y_n)$ such that for each i independently, coordinate Y_i is picked uniformly in $\{X_i, X_i + 1 \bmod 3\}$. Our goal is to show that:

$$\Pr[\underline{X} \in S \wedge \underline{Y} \in S] \geq c(\mu) > 0\ .$$

In other words, we want to bound away the probability from 0 by an expression which only depends on μ and not on n.

* Part of this work was done while T. H. and E. M. were at the Simons Institute.
† J. H was supported by the Swiss National Science Foundation (SNF), project no. 200021-132508.
‡ E. M. was supported by NSF grant DMS-1106999, NSF Grant CCF 1320105 and DOD ONR grant N000141110140 and grant 328025 from the Simons foundation.


Distributed according to $\underline{\mathcal{P}} := \mathcal{P}^n$.
Tuples $\overline{X}_i$ are i.i.d. according to $\mathcal{P}$. Each of the ℓ marginals of $\mathcal{P}$ is π.
Vectors $\underline{X}^{(j)}$ are distributed (dependently) according to $\underline{\pi} := \pi^n$.
$\underline{\overline{X}}$
$\overline{X}_1$ $\overline{X}_2$ $\cdots$ $\overline{X}_i$ $\cdots$ $\overline{X}_n$
$\underline{X}^{(1)}$ $X_1^{(1)}$ $X_2^{(1)}$ $\cdots$ $X_i^{(1)}$ $\cdots$ $X_n^{(1)}$
$\underline{X}^{(2)}$ $X_1^{(2)}$ $X_2^{(2)}$ $\cdots$ $X_i^{(2)}$ $\cdots$ $X_n^{(2)}$
$\underline{X}^{(j)}$ $X_1^{(j)}$ $X_2^{(j)}$ $\cdots$ $X_i^{(j)}$ $\cdots$ $X_n^{(j)}$
$\underline{X}^{(\ell)}$ $X_1^{(\ell)}$ $X_2^{(\ell)}$ $\cdots$ $X_i^{(\ell)}$ $\cdots$ $X_n^{(\ell)}$

$$\alpha(\mathcal{P}) := \min_{x \in \Omega} \mathcal{P}(x, x, \ldots, x)$$
$$\rho(\mathcal{P}) : \text{See Definition 10}$$

$$\begin{aligned} X_i^{(j)} &\in \Omega \\ \underline{X}^{(j)} &\in \underline{\Omega} := \Omega^n \\ \overline{X}_i &\in \overline{\Omega} := \Omega^\ell \\ \underline{\overline{X}} &\in \underline{\overline{\Omega}} := \Omega^{n \cdot \ell} \\ S &\subseteq \underline{\Omega} \end{aligned}$$

Figure 1 Naming of the random variables in the general case. The columns $\overline{X}_i$ are distributed *i.i.d* according to $\mathcal{P}$. Each $X_i^{(j)}$ is distributed according to π. The overall distribution of $\underline{\overline{X}}$ is $\underline{\mathcal{P}}$.

1.2 Our results

More generally, let Ω be a finite alphabet and assume we are given a probability distribution $\mathcal{P}$ over Ω^ℓ for some $\ell \geq 2$ – we will call it an *ℓ-step probability distribution over* Ω.

Furthermore, assume we are given $n \in \mathbb{N}$. We consider ℓ vectors $\underline{X}^{(1)}, \ldots, \underline{X}^{(\ell)}$, $\underline{X}^{(j)} = (X_1^{(j)}, \ldots, X_n^{(j)})$ such that for every $i \in [n]$, the ℓ-tuple $(X_i^{(1)}, \ldots, X_i^{(\ell)})$ is sampled according to $\mathcal{P}$, independently of the other coordinates $i' \neq i$ (see Figure 1 for an overview of the notation).

▶ **Definition 1.** Let $\mu, \delta \in (0, 1]$. We say that a distribution $\mathcal{P}$ *is* (μ, δ)*-same-set hitting*, if, whenever a function $f : \Omega^n \to [0, 1]$ satisfies $\mathrm{E}[f(\underline{X}^{(j)})] \geq \mu$ for every $j \in [\ell] := \{1, \ldots, \ell\}$, we have

$$\mathrm{E}\left[\prod_{j=1}^{\ell} f(\underline{X}^{(j)})\right] \geq \delta\,.$$

We call $\mathcal{P}$ *same-set hitting* if for every $\mu \in (0, 1]$ there exists $\delta \in (0, 1]$ such that $\mathcal{P}$ is (μ, δ)-same-set hitting.

In this paper we address the question: which distributions $\mathcal{P}$ are same-set hitting? We achieve full characterisation for $\ell = 2$ and answer the question affirmatively for a large class of distributions with $\ell > 2$.

To explain related work and our results, we introduce a stronger notion:

▶ **Definition 2.** Let $\mu, \delta \in (0,1]$. We say that a distribution $\mathcal{P}$ *is* (μ,δ)*-set hitting*, if, whenever functions $f^{(1)}, \dots, f^{(\ell)} : \Omega^n \to [0,1]$ satisfy $\mathrm{E}[f^{(j)}(\underline{X}^{(j)})] \ge \mu$ for every $j \in [\ell]$, we have

$$\mathrm{E}\left[\prod_{j=1}^{\ell} f^{(j)}(\underline{X}^{(j)})\right] \ge \delta \; . \tag{1}$$

We call $\mathcal{P}$ *set hitting* if for every $\mu \in (0,1]$ there exists $\delta \in (0,1]$ such that $\mathcal{P}$ is (μ,δ)-set hitting.

The full classification of set hitting distributions can be deduced from a paper on reverse hypercontractivity[1] by Mossel, Oleszkiewicz and Sen [10]:

▶ **Theorem 3** (follows from [10])**.** *A probability space $\mathcal{P}$ is set hitting if and only if:*

$$\min_{\substack{x^{(1)} \in \mathrm{supp}(X_i^{(1)}), \\ \dots, \\ x^{(\ell)} \in \mathrm{supp}(X_i^{(\ell)})}} \mathcal{P}(x^{(1)}, \dots, x^{(\ell)}) > 0 \; . \tag{2}$$

To state our results, we need to introduce the following properties of $\mathcal{P}$:

▶ **Definition 4.** We say that $\mathcal{P}$ *has equal marginals* if for every $j \in [\ell]$ and every $x \in \Omega$:

$$\Pr[X_i^{(1)} = x] = \ldots = \Pr[X_i^{(j)} = x] = \ldots = \Pr[X_i^{(\ell)} = x] \; .$$

As explained in Section 3.4.2, the same-set hitting is interesting only for distributions with equal marginals. Whenever we discuss such distributions, we assume w.l.o.g that Ω is equal to the support of the marginal.

▶ **Definition 5.** We define:

$$\alpha(\mathcal{P}) := \min_{x \in \Omega} \mathcal{P}(x, \dots, x) \; ,$$
$$\beta(\mathcal{P}) := \min_{x^{(1)}, \dots, x^{(\ell)} \in \Omega} \mathcal{P}(x^{(1)}, \dots, x^{(\ell)}) \; .$$

1.2.1 The case of two steps

In case of $\ell = 2$ we establish the following theorem:

▶ **Theorem 6** (cf. Theorem 12)**.** *A two-step probability distribution with equal marginals $\mathcal{P}$ is same-set hitting if and only if $\alpha(\mathcal{P}) > 0$.*

Of course, if $\beta(\mathcal{P}) > 0$, then Theorem 6 follows from Theorem 3. However, we are not aware of any previous work in case $\beta(\mathcal{P}) = 0$, i.e., when the distribution is same-set hitting but not set hitting, in particular for the probability space from Section 1.1.

[1] That $\mathcal{P}$ is set hitting if (2) holds is a consequence of Lemma 8.3 in [10]. If (2) does not hold, an appropriate combination of dictators establishes a counterexample.

1.2.2 More than two steps

In a general case of an ℓ-step distribution with equal marginals, it is still clear that $\alpha(\mathcal{P}) > 0$ is necessary. However, it remains open if it is sufficient.

We provide the following partial results. Firstly, by a simple inductive argument based on Theorem 12, we show that multi-step probability spaces induced by Markov chains are same-set hitting.

Secondly, we show that $\mathcal{P}$ is same-set hitting if $\alpha(\mathcal{P}) > 0$ and its *correlation* $\rho(\mathcal{P})$ is smaller than 1. The opposite condition $\rho(\mathcal{P}) = 1$ is equivalent to the following: There exist $j \in [\ell]$, $S \subseteq \Omega$, $T \subseteq \Omega^{\ell-1}$ such that $0 < |S| < |\Omega|$ and:

$$X_i^{(j)} \in S \iff \left(X_i^{(1)}, \dots, X_i^{(j-1)}, X_i^{(j+1)}, \dots, X_i^{(\ell)}\right) \in T \ .$$

For the full definition of $\rho(\mathcal{P})$, cf. Definition 10.

▶ **Theorem 7** (cf. Theorem 13). *Let $\mathcal{P}$ be a probability distribution with equal marginals. If $\alpha(\mathcal{P}) > 0$ and $\rho(\mathcal{P}) < 1$, then $\mathcal{P}$ is same-set hitting.*

We are not aware of any general results in case $\rho(\mathcal{P}) = 1$. In particular, let $\mathcal{P}$ be a three-step distribution over $\Omega = \{0, 1, 2\}$ such that $X_i^{(1)}, X_i^{(2)}, X_i^{(3)}$ is uniform over $\{000, 111, 222, 012, 120, 201\}$. To the best of our knowledge, it is an open question whether this distribution $\mathcal{P}$ is same-set hitting.

1.2.3 Set hitting for functions with no large Fourier coefficients

The methods developed here also allow to obtain lower bounds on the probability of hitting multiple sets. In fact, we show that if $\rho(\mathcal{P}) < 1$, then such lower bounds exist in terms of ρ, the measures of the sets and the largest non-empty Fourier coefficient.

▶ **Theorem 8** (Informal, cf. Theorem 14). *Let $\mathcal{P}$ be a probability distribution with $\rho(\mathcal{P}) < 1$. Then, $\mathcal{P}$ is set-hitting for functions $f^{(1)}, \dots, f^{(\ell)} : \underline{\Omega} \to [0, 1]$ that have both:*

- *Noticeable expectations, i.e.,* $\mathrm{E}[f^{(j)}(\underline{X}^{(j)})] \geq \Omega(1)$.
- *No large Fourier coefficients, i.e.,* $\max_\sigma \left|\hat{f}^{(j)}(\sigma)\right| \leq o(1)$.

1.3 Background and related work

The question of understanding which sets are hit often by dynamical systems is central to ergodic theory and additive combinatorics.

The Poincaré recurrence theorem states that measure preserving maps satisfy the property that for each set U of positive measure, and for almost every point x of U, the dynamical system started at x will hit U infinitely often, see, e.g., [12].

Much of the ergodic theory deals with quantifying this phenomena of repeatedly hitting the set U. The ergodic theorem, for example, implies that ergodic measure preserving dynamical systems satisfy that for almost all starting points x, the set U will be hit in limiting frequency which is equal to its measure.

Understanding set hitting by a number of consecutive steps of a process has been of intense study in additive combinatorics (where almost always $\rho = 1$).

For example, a well-studied case are random arithmetic progressions. Let Z be a finite additive group and $\ell \in \mathbb{N}$. Then, we can define a distribution $\mathcal{P}_{Z,\ell}$ of random ℓ-step arithmetic progressions in Z. Specifically, for every $x, r \in Z$ we set:

$$\mathcal{P}_{Z,\ell}(x, x + r, x + 2r, \dots, x + (\ell - 1)r) := 1/|Z|^2 \ .$$

Some of the distributions $\mathcal{P}_{Z,\ell}$ can be shown to be same-set hitting using the hypergraph regularity lemma:

▶ **Theorem 9** ([14], [15], [6], cf. Theorem 11.27, Proposition 11.28 and Exercise 11.6.3 in [17]). *If $|Z|$ is coprime to $(\ell-1)!$, then $\mathcal{P}_{Z,\ell}$ is same-set hitting.*

This follows a long line of work, starting by Szemerédi lemma [16], its proof by Furstenberg using the ergodic theorem [3] as well as the finite group and multi-dimensional versions, see, e.g., [13, 4, 5].

One might conjecture that $\alpha(\mathcal{P}) > 0$ is the sole sufficient condition for same-set hitting. Unfortunately, the techniques used to prove Theorem 9 do not seem to extend easily to less symmetric spaces. This suggests that proving the conjecture fully in $\rho = 1$ case might be a difficult undertaking.

The case of $\rho < 1$ has also been studied in the context of extremal combinatorics and hardness of approximation. In particular, Mossel [9] uses the invariance principle to prove that if $\rho(\mathcal{P}) < 1$, then $\mathcal{P}$ is set hitting for low-influence functions. We use this result to establish Theorem 7. Additionally, Theorem 8 can be seen as a strengthening of [9].

Furthermore, Austrin and Mossel [1] establish the result equivalent to Theorem 8 assuming in addition to $\rho(\mathcal{P}) < 1$ also that $\mathcal{P}$ is pairwise independent (they also prove results for the case $\rho(\mathcal{P}) = 1$ with pairwise independence but these involve only bounded degree functions).

Finally we note that another relevant paper in the case of $\ell = 2$ and symmetric $\mathcal{P}$ is by Dinur, Friedgut and Regev [2], who give a characterization of non-hitting sets. However, due to a different framework, their results are not directly comparable to ours.

We hope that our work might turn out to be useful in inapproximability. For example, our theorem is related to the proof of hardness for rainbow colorings of hypergraphs by Guruswami and Lee [7]. In particular, it is connected to their Theorem 4.3 and partially answers their Questions C.4 and C.6.

1.4 Outline of the paper

The rest of the paper is organised as follows: the notation is introduced in Section 2, Section 3 contains full statements of our theorems and Section 4 sketches the proof of our main theorem.

The full proofs along with he modified proof of the low-influence theorem from [9] can be found in the Arxiv version of the paper [8].

2 Notation and Preliminaries

2.1 Notation

We will now introduce our setting and notation. We refer the reader to Figure 1 for an overview.

We always assume that we have n independent coordinates. In each coordinate i we pick ℓ values $X_i^{(j)}$ for $j \in [\ell] = \{1, \ldots, \ell\}$ at random using some distribution. Each value $X_i^{(j)}$ is chosen from the same fixed set Ω, and the distribution of the tuple $\overline{X}_i = (X_i^{(1)}, \ldots, X_i^{(\ell)})$ of values from Ω^ℓ is given by a distribution $\mathcal{P}$.

This gives us values $X_i^{(j)}$ for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, \ell\}$. Thus, we have ℓ vectors $\underline{X}^{(1)}, \ldots, \underline{X}^{(\ell)}$, where $\underline{X}^{(j)} = (X_1^{(j)}, \ldots, X_n^{(j)})$ represents the j-th step of the random process. In case $\ell = 2$, we might call our two vectors $\underline{X}$ and $\underline{Y}$ instead.

For reasons outlined in Section 3.4.2 we assume that all of $X_i^{(1)}, \ldots, X_i^{(\ell)}$ have the same marginal distribution, which we call π. We assume that Ω is the support of π.

Even though it is not necessary, for clarity of the presentation we assume that each coordinate $\overline{X}_i = (X_i^{(1)}, \ldots, X_i^{(j)}, \ldots, X_i^{(\ell)})$ has the same distribution $\mathcal{P}$.

We consistently use index i to index over the coordinates (from $[n]$) and j to index over the steps (from $[\ell]$).

As visible in Figure 1, we denote the aggregation across the coordinates by the underline and the aggregation across the steps by the overline. For example, we write $\underline{\Omega} = \Omega^n$, $\overline{\Omega} = \Omega^\ell$, $\underline{\mathcal{P}} = \mathcal{P}^n$ and $\underline{\overline{X}} = (\overline{X}_1, \ldots, \overline{X}_n) = (\underline{X}^{(1)}, \ldots, \underline{X}^{(\ell)})$.

We sometimes call $\underline{\mathcal{P}}$ a tensorized, multi-step probability distribution as opposed to a tensorized, single-step distribution $\underline{\pi}$ and single-coordinate, multi-step distribution $\mathcal{P}$.

Furthermore, we extend the index notation to subsets of indices or steps. For example, for $S \subseteq [\ell]$ we define $X^{(S)}$ to be the collection of random variables $\left\{X^{(j)} : j \in S\right\}$.

We also use the set difference symbol to mark vectors with one element missing, e.g., $\overline{X}^{\setminus j} := (X^{(1)}, \ldots, X^{(j-1)}, X^{(j+1)}, \ldots, X^{(\ell)})$.

One should think of ℓ and $|\Omega|$ as constants and of n as large. We aim to get bounds which are independent of n.

2.2 Correlation

In case $\ell > 2$, the bound we obtain will depend on the *correlation* of the distribution $\mathcal{P}$. This concept was used before in [9].

▶ **Definition 10.** Let $\mathcal{P}$ be a single-coordinate distribution and let $S, T \subseteq [\ell]$. We define the *correlation*:

$$\rho(\mathcal{P}, S, T) := \sup\Big\{\mathrm{Cov}[f(X^{(S)}), g(X^{(T)})] \;\Big|\; f : \Omega^{(S)} \to \mathbb{R}, g : \Omega^{(T)} \to \mathbb{R}, \\ \mathrm{Var}[f(X^{(S)})] = \mathrm{Var}[g(X^{(T)})] = 1\Big\} .$$

The correlation of $\mathcal{P}$ is $\rho(\mathcal{P}) := \max_{j \in [\ell]} \rho\left(\mathcal{P}, \{j\}, [\ell] \setminus \{j\}\right)$.

2.3 Influence

A crucial notion in the proof of Theorem 7 is the *influence* of a function. It expresses the average variance of a function, given that all but one of its n inputs have been fixed to random values:

▶ **Definition 11.** Let $\underline{X}$ be a random vector over alphabet $\underline{\Omega}$ and $f : \underline{\Omega} \to \mathbb{R}$ be a function and $i \in [n]$. The *influence of f on the i-th coordinate* is:

$$\mathrm{Inf}_i(f(\underline{X})) := \mathrm{E}\left[\mathrm{Var}\left[f(\underline{X}) \mid \underline{X}_{\setminus i}\right]\right] .$$

The *(total) influence of f* is $\mathrm{Inf}(f(\underline{X})) := \sum_{i=1}^n \mathrm{Inf}_i(f(\underline{X}))$.

Note that the influence depends both on the function f and the distribution of the vector $\underline{X}$.

3 Our Results

Here we give precise statements of our results presented in the introduction.

3.1 The case of $\ell = 2$

▶ **Theorem 12.** *Let Ω be a finite set and $\mathcal{P}$ a probability distribution over Ω^2 with equal marginals π. Let pairs (X_i, Y_i) be i.i.d. according to $\mathcal{P}$ for $i \in \{1, \dots, n\}$.*

Then, for every $f : \Omega^n \to [0,1]$ with $\mathrm{E}[f(\underline{X})] = \mu > 0$:

$$\mathrm{E}[f(\underline{X})f(\underline{Y})] \geq c\left(\alpha(\mathcal{P}), \mu\right) \ , \tag{3}$$

where the function $c()$ is positive whenever $\alpha(\mathcal{P}) > 0$.

We remark that Theorem 12 does not depend on $\rho(\mathcal{P})$ in any way. This is in contrast to the case $\ell > 2$. It is possible to obtain a polynomially large explicit bound $c()$ for symmetric two-step spaces.

To prove Theorem 12 we make a convex decomposition argument and then apply the multi-step Theorem 13. For completeness, we provide a proof of Theorem 6 assuming Theorem 12.

Proof of Theorem 6. The "if" part follows from Theorem 12. The "only if" can be seen by taking f to be an appropriate dictator. ◀

3.2 The general case

▶ **Theorem 13.** *Let Ω be a finite set and $\mathcal{P}$ a distribution over Ω^ℓ in which all marginals are equal. Let tuples $\overline{X}_i = (X_i^{(1)}, \dots, X_i^{(\ell)})$ be i.i.d. according to $\mathcal{P}$ for $i \in \{1, \dots, n\}$.*

Then, for every function $f : \Omega^n \to [0,1]$ with $\mathrm{E}[f(\underline{X}^{(j)})] = \mu > 0$:

$$\mathrm{E}\left[\prod_{j=1}^{\ell} f(\underline{X}^{(j)})\right] \geq c\left(\alpha(\mathcal{P}), \rho(\mathcal{P}), \ell, \mu\right) \ , \tag{4}$$

where the function $c()$ is positive whenever $\alpha(\mathcal{P}) > 0$ and $\rho(\mathcal{P}) < 1$.

Furthermore, there exists some $D(\mathcal{P}) > 0$ (more precisely, D depends on α, ρ and ℓ) such that if $\mu \in (0, 0.99]$, one can take:

$$c(\alpha, \rho, \ell, \mu) := 1/\exp\left(\exp\left(\exp\left((1/\mu)^D\right)\right)\right) \ . \tag{5}$$

Note that this bound *does* depend on $\rho(\mathcal{P})$. We also obtain a bound that does not depend on $\rho(\mathcal{P})$ for multi-step probability spaces generated by Markov chains.

3.3 Hitting of different sets by uniform functions

Finally, we state the generalization of low-influence theorem from [9]. We assume that the reader is familiar with Fourier coefficients $\hat{f}(\sigma)$ and the basics of discrete function analysis, for details see, e.g., Chapter 8 of [11]. For the proof see the full version of the paper [8].

▶ **Theorem 14.** *Let $\overline{X}$ be a random vector distributed according to an ℓ-step distribution $\mathcal{P}$ with $\rho(\mathcal{P}) \leq \rho < 1$ and let $\mu^{(1)}, \dots, \mu^{(\ell)} \in (0,1]$.*

There exist $k \in \mathbb{N}$ and $\gamma > 0$ (both depending only on $\mathcal{P}$ and $\mu^{(1)}, \dots, \mu^{(\ell)}$) such that for all functions $f^{(1)}, \dots, f^{(\ell)} : \underline{\Omega} \to [0,1]$, if $\mathrm{E}[f^{(j)}(\underline{X}^{(j)})] = \mu^{(j)}$ and $\max_{\sigma: 0 < |\sigma| \leq k} |\hat{f}^{(j)}(\sigma)| \leq \gamma$, then

$$\mathrm{E}\left[\prod_{j=1}^{\ell} f^{(j)}(\underline{X}^{(j)})\right] \geq c(\mathcal{P}, \mu^{(1)}, \dots, \mu^{(\ell)}) > 0 \ . \tag{6}$$

3.4 Assumptions of the theorems

3.4.1 Equal distributions: unnecessary

In Theorems 12, 13 and 14 we assumed that the tuples $(X_i^{(1)}, \dots, X_i^{(\ell)})$ are distributed identically for each i. It is natural to ask if it is indeed necessary.

This is not the case. Instead, we made this assumption for simplicity of notation and presentation. If one is interested in statements which are valid where coordinate i is distributed according to $\mathcal{P}_i$, one simply needs to assume that there are $\alpha > 0$ and $\rho < 1$ such that $\alpha(\mathcal{P}_i) \geq \alpha$ and $\rho(\mathcal{P}_i) \leq \rho$.

3.4.2 Equal marginals: necessary

We quickly discuss the case when $\mathcal{P}$ does not have equal marginals. Recall that $\beta(\mathcal{P}) = \min_{x^{(1)},\dots,x^{(\ell)} \in \Omega} \mathcal{P}(x^{(1)}, \dots, x^{(\ell)})$. If $\beta(\mathcal{P}) > 0$, then, by Theorem 3, $\mathcal{P}$ is set hitting, and therefore also same-set hitting.

In case $\beta(\mathcal{P}) = 0$, we demonstrate an example which shows that $\mathrm{E}\left[\prod_{j=1}^{\ell} f(\underline{X}^{(j)})\right]$ can be exponentially small in n. For concreteness, we set $\ell := 2$ and $\Omega := \{0, 1\}$ and consider $\mathcal{P}$ which picks uniformly among $\{00, 01, 11\}$. We then set

$$S_1 := \{(x_1, \dots, x_n) \mid x_1 = 1 \wedge |\mathrm{wt}(x) - n/3| \leq 0.01n\} \tag{7}$$

$$S_2 := \{(x_1, \dots, x_n) \mid x_1 = 0 \wedge |\mathrm{wt}(x) - 2n/3| \leq 0.01n\} \tag{8}$$

where $\mathrm{wt}(x)$ is the Hamming-weight of x, i.e., the number of ones in x.

For large enough n, a concentration bound implies that $\Pr[\underline{X}^{(1)} \in S_1] > \frac{1}{3} - 0.01$ and $\Pr[\underline{X}^{(2)} \in S_2] > \frac{1}{3} - 0.01$. Hence, if we set f to be the indicator function of $S := S_1 \cup S_2$, the assumption of Theorem 13 holds. However, because of the first coordinate we have $\Pr[\underline{X}^{(1)} \in S \wedge \underline{X}^{(2)} \in S] \leq \Pr[\underline{X}^{(1)} \in S_2 \vee \underline{X}^{(2)} \in S_1]$, and the right hand side is easily seen to be exponentially small.

It is not difficult to extend this example to any distribution with $\beta(\mathcal{P}) = 0$ that does not have equal marginals.

4 Proof Sketch

In this section we briefly outline the proof of Theorem 13. For simplicity, we assume that the probability space is the one from Section 1.1, i.e., (X_i, Y_i) are distributed uniformly in $\{00, 11, 22, 01, 12, 20\}$. Additionally, we assume that we are given a set $S \subseteq \{0, 1, 2\}^n$ with $\mu(S) = |S|/3^n > 0$, so that we want a bound of the form

$$\Pr\left[\underline{X} \in S \wedge \underline{Y} \in S\right] \geq c(\mu) > 0 \,.$$

The proof consists of three steps. Intuitively, in the first step we deal with dictator sets, e.g., $S_{\mathrm{dict}} = \{\underline{x} : x_1 = 0\}$, in the second step with linear sets, e.g., $S_{\mathrm{lin}} = \{\underline{x} : \sum_{i=1}^{n} x_i \pmod 3 = 0\}$ and in the third step with threshold sets, e.g., $S_{\mathrm{thr}} = \{\underline{x} : |\{i : x_i = 0\}| \geq n/3\}$.

4.1 Step 1 – making a set resilient

We call a set resilient if $\Pr[\underline{X} \in S]$ does not change by more than a (small) multiplicative constant factor whenever conditioned on $(X_{i_1} = x_{i_1}, \dots, X_{i_s} = x_{i_s})$ on a constant number s of coordinates.

In particular, S_{dict} is not resilient (because conditioning on $x_1 = 0$ increases the measure of the set to 1), while S_{lin} and S_{thr} are.

If a set is not resilient, using $\mathcal{P}(x,x) = 1/6$ for every $x \in \Omega$, one can find an event $\mathcal{E} :\equiv X_{i_1} = Y_{i_1} = x_{i_1} \land \ldots \land X_{i_s} = Y_{i_s} = x_{i_s}$ such that for some constant $\epsilon > 0$ we have $\Pr[\mathcal{E}] \ge \epsilon$ and, at the same time, $\Pr[\underline{X} \in S \mid \mathcal{E}] \ge (1+\epsilon)\Pr[\underline{X} \in S]$.

Since each such conditioning increases the measure of the set S by a constant factor, S must become resilient after a constant number of its iterations. Furthermore, each conditioning induces only a constant factor loss in $\Pr[\underline{X} \in S \land \underline{Y} \in S]$.

4.2 Step 2 – eliminating high influences

In this step, assuming that S is resilient, we condition on a constant number of coordinates to transform it into two sets S' and T' such that:

- Both of them have low influences on all coordinates.
- Both of them are supersets of S (after the conditioning).

The first property allows us to apply low-influence set hitting from [9] to S' and T'. The second one, together with the resilience of S, ensures that $\mu(S'), \mu(T') \ge (1-\epsilon)\mu(S)$.

In fact, it is more convenient to assume that we are initially given two resilient sets S and T.

Assume w.l.o.g. that $\text{Inf}_i(T) \ge \tau$ for some $i \in [n]$. Given $z \in \{0,1,2\}$, let $T_z := \{\underline{x} : \underline{x} \in T \land x_i = z\}$. Furthermore, let $T_z^* := T_z \cup T_{z+1 \pmod 3}$.

Since $\text{Inf}_i(T) \ge \tau$, we can show that there exists $z \in \{0,1,2\}$ such that, after conditioning on $X_i = Y_i = z$, the sum $\mu(S_z) + \mu(T_z^*)$ is strictly greater than the sum $\mu(S) + \mu(T)$:

$$\Pr[\underline{X} \in S_z \mid X_i = z] + \Pr[\underline{Y} \in T_z^* \mid Y_i = z] \ge \Pr[\underline{X} \in S] + \Pr[\underline{Y} \in T] + c(\tau) \ . \tag{9}$$

We choose to delete the coordinate i and replace S with $S' := S_z$ and T with $T' := T_z^*$. Equation (9) implies that after a constant number of such operations, neither S nor T has any remaining high-influence coordinates.

Crucially, with respect to same-set hitting our set replacement is essentially equivalent to conditioning on $X_i = z$ and $Y_i = z \lor Y_i = z+1 \pmod 3$. Therefore, each operation induces only a constant factor loss in $\Pr[\underline{X} \in S \land \underline{Y} \in T]$.

4.3 Step 3 – applying low-influence theorem from [9]

Once we are left with two low-influence, somewhat-large sets S and T, we obtain $\Pr[\underline{X} \in S \land \underline{Y} \in T] \ge c(\mu) > 0$ by a straightforward application of a slightly modified version of Theorem 1.14 from [9]. The theorem gives that $\rho(\mathcal{P}) < 1$ implies that the distribution $\mathcal{P}$ is set hitting for low-influence functions:

▶ **Theorem 15.** *Let $\overline{\underline{X}}$ be a random vector distributed according to $(\underline{\Omega}, \underline{\mathcal{P}})$ such that $\mathcal{P}$ has equal marginals, $\rho(\mathcal{P}) \le \rho < 1$ and $\min_{x \in \Omega} \pi(x) \ge \alpha > 0$.*

Then, for all $\epsilon > 0$, there exists $\tau := \tau(\epsilon, \rho, \alpha, \ell) > 0$ such that if functions $f^{(1)}, \ldots, f^{(\ell)} : \underline{\Omega} \to [0,1]$ satisfy

$$\max_{i \in [n], j \in [\ell]} \text{Inf}_i(f^{(j)}(\underline{X}^{(j)})) \le \tau \ , \tag{10}$$

then, for $\mu^{(j)} := \text{E}[f^{(j)}(\underline{X}^{(j)})]$:

$$\text{E}\left[\prod_{j=1}^{\ell} f^{(j)}(\underline{X}^{(j)})\right] \ge \left(\prod_{j=1}^{\ell} \mu^{(j)}\right)^{\ell/(1-\rho^2)} - \epsilon \ . \tag{11}$$

Furthermore, there exists an absolute constant $C \geq 0$ such that for $\epsilon \in (0, 1/2]$ one can take

$$\tau := \left(\frac{(1-\rho^2)\epsilon}{\ell^{5/2}}\right)^{C^{\frac{\ell \ln(\ell/\epsilon)\ln(1/\alpha)}{(1-\rho)\epsilon}}} . \tag{12}$$

The proof of Theorem 15 can be found in the Arxiv version of the paper [8].

4.4 The case $\rho = 1$: open question

Theorem 13 requires that $\rho < 1$ in order to give a meaningful bound. It is unclear whether this is an artifact of our proof or if it is necessary. In particular, consider the three step distribution $\mathcal{P}$ which picks a uniform triple from $\{000, 111, 222, 012, 120, 201\}$. One easily checks that $\rho(\mathcal{P}) = 1$ and that all marginals are uniform. We do not know if this distribution is same-set hitting.

However, the method of our proof breaks down. We illustrate the reason in the following lemma.

▶ **Lemma 16.** *For every $n > n_0$ there exist three sets $S^{(1)}$, $S^{(2)}$, and $S^{(3)}$ such that for the distribution $\mathcal{P}$ as described above we have*

- $\forall j : \Pr[\underline{X}^{(j)} \in S^{(j)}] \geq 0.49$.
- $\Pr[\forall j : \underline{X}^{(j)} \in S^{(j)}] = 0$.
- *The characteristic functions $1_{S^{(j)}}$ of the three sets all satisfy*

$$\max_{i \in [n]} \mathrm{Inf}_i(1_{S^{(j)}}(\underline{X}^{(j)})) \to 0 \text{ as } n \to \infty .$$

While the lemma does not give information about whether $\mathcal{P}$ is same-set hitting, it shows that our proof fails (since the analogue of Theorem 15 fails).

Proof. We let

$$S^{(1)} := \{\underline{x}^{(1)} : \underline{x}^{(1)} \text{ has less than } n/3 \text{ twos}\} ,$$
$$S^{(2)} := \{\underline{x}^{(2)} : \underline{x}^{(2)} \text{ has less than } n/3 \text{ ones}\} ,$$
$$S^{(3)} := \{\underline{x}^{(3)} : \underline{x}^{(3)} \text{ has less than } n/3 \text{ zeros}\} .$$

Whenever we pick $\underline{X}^{(1)}, \underline{X}^{(2)}, \underline{X}^{(3)}$, the number of twos in $\underline{X}^{(1)}$ plus the number of ones in $\underline{X}^{(2)}$ plus the number of zeros in $\underline{X}^{(3)}$ always equals n (there is a contribution of one from each coordinate). All three properties are now easy to check. ◀

References

1 Per Austrin and Elchanan Mossel. Noise correlation bounds for uniform low degree functions. *Arkiv för Matematik*, 51(1):29–52, 2013. doi:10.1007/s11512-011-0145-5.

2 Irit Dinur, Ehud Friedgut, and Oded Regev. Independent sets in graph powers are almost contained in juntas. *Geometric and Functional Analysis*, 18(1):77–97, 2008.

3 Harry Furstenberg. Ergodic behavior of diagonal measures and a theorem of Szemerédi on arithmetic progressions. *Journal d'Analyse Mathématique*, 31(1):204–256, 1977. doi:10.1007/BF02813304.

4 Harry Furstenberg and Yitzhak Katznelson. A density version of the Hales-Jewett theorem. *Journal d'Analyse Mathématique*, 57(1):64–119, 1991. doi:10.1007/BF03041066.

5 W. T. Gowers. A new proof of Szemerédi's theorem. *Geometric & Functional Analysis GAFA*, 11(3):465–588, 2001. `doi:10.1007/s00039-001-0332-9`.

6 W. T. Gowers. Hypergraph regularity and the multidimensional Szemerédi theorem. *Annals of Mathematics*, 166(3):897–946, 2007. Available from: `http://www.jstor.org/stable/20160083`.

7 Venkatesan Guruswami and Euiwoong Lee. Strong inapproximability results on balanced rainbow-colorable hypergraphs. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 822–836, 2015. `doi:10.1137/1.9781611973730.56`.

8 Jan Hązła, Thomas Holenstein and Elchanan Mossel. Lower Bounds on Same-Set Inner Product in Correlated Spaces, 2015. Arxiv `https://arxiv.org/abs/1509.06191`.

9 Elchanan Mossel. Gaussian bounds for noise correlation of functions. *Geometric and Functional Analysis*, 19(6):1713–1756, 2010. `doi:10.1007/s00039-010-0047-x`.

10 Elchanan Mossel, Krzysztof Oleszkiewicz, and Arnab Sen. On reverse hypercontractivity. *Geometric and Functional Analysis*, 23(3):1062–1097, 2013. `doi:10.1007/s00039-013-0229-4`.

11 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. Available from: `http://www.cambridge.org/de/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/analysis-boolean-functions`.

12 Karl Petersen. *Ergodic theory*. Cambridge University Press, corrected edition, 1989. Available from: `http://opac.inria.fr/record=b1088586`.

13 Klaus F. Roth. On certain sets of integers. *Journal of the London Mathematical Society*, s1-28(1):104–109, 1953. `doi:10.1112/jlms/s1-28.1.104`.

14 Vojtěch Rödl and Jozef Skokan. Regularity lemma for k-uniform hypergraphs. *Random Structures & Algorithms*, 25(1):1–42, 2004. `doi:10.1002/rsa.20017`.

15 Vojtěch Rödl and Jozef Skokan. Applications of the regularity lemma for uniform hypergraphs. *Random Structures & Algorithms*, 28(2):180–194, 2006. `doi:10.1002/rsa.20108`.

16 Endre Szemerédi. On sets of integers containing no k elements in arithmetic progression. *Acta Arithmetica*, 27(1):199–245, 1975.

17 Terence Tao and Van H. Vu. *Additive Combinatorics*. Cambridge University Press, 2006. Available from: `http://opac.inria.fr/record=b1122796`.

Estimating Parameters Associated with Monotone Properties*

Carlos Hoppen[1], Yoshiharu Kohayakawa[2], Richard Lang[3], Hanno Lefmann[4], and Henrique Stagni[5]

1 **Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil**
`choppen@ufrgs.br`

2 **Universidade de São Paulo, São Paulo, Brazil**
`yoshi@ime.usp.br`

3 **Universidad de Chile, Santiago, Chile**
`rlang@dim.uchile.cl`

4 **Technische Universität Chemnitz, Chemnitz, Germany**
`Lefmann@Informatik.TU-Chemnitz.de`

5 **Universidade de São Paulo, São Paulo, Brazil**
`stagni@ime.usp.br`

Abstract

There has been substantial interest in estimating the value of a graph parameter, i.e., of a real function defined on the set of finite graphs, by sampling a randomly chosen substructure whose size is independent of the size of the input. Graph parameters that may be successfully estimated in this way are said to be *testable* or *estimable*, and the *sample complexity* $q_z = q_z(\varepsilon)$ of an estimable parameter z is the size of the random sample required to ensure that the value of $z(G)$ may be estimated within error ε with probability at least $2/3$. In this paper, we study the sample complexity of estimating two graph parameters associated with a monotone graph property, improving previously known results. To obtain our results, we prove that the vertex set of any graph that satisfies a monotone property $\mathcal{P}$ may be partitioned equitably into a constant number of classes in such a way that the cluster graph induced by the partition is not far from satisfying a natural weighted graph generalization of $\mathcal{P}$. Properties for which this holds are said to be *recoverable*, and the study of recoverable properties may be of independent interest.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases parameter estimation, parameter testing, edit distance to monotone graph properties, entropy of subgraph classes, speed of subgraph classes

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.35

1 Introduction

In the last two decades, a lot of effort has been put into finding constant-time randomized algorithms (conditional on sampling) to gauge whether a combinatorial structure satisfies

* C. Hoppen acknowledges the support of FAPERGS (Proc. 2233-2551/14-0) and CNPq (Proc. 448754/2014-2 and 308539/2015-0). C. Hoppen and H. Lefmann acknowledge the support of CAPES and DAAD via PROBRAL (CAPES Proc. 408/13 and DAAD 56267227 and 57141126 and 57245206). C. Hoppen, Y. Kohayakawa and H. Stagni thank FAPESP (Proc. 2013/03447-6) and NUMEC/USP (Project MaCLinC/USP) for their support. Y. Kohayakawa was partially supported by FAPESP (2013/07699-0) and CNPq (310974/2013-5 and 459335/2014-6). H. Stagni was supported by FAPESP (2015/15986-4) and CNPq (141970/2015-4 and 459335/2014-6).

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016).
Editors: Klaus Jansen, Claire Matthieu, José D. P. Rolim, and Chris Umans; Article No. 35; pp. 35:1–35:13
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

some property, or to estimate the value of some numerical function associated with this combinatorial structure. In this paper, we focus on the graph case and, as usual, we consider algorithms that have the ability to query whether any desired pair of vertices in the input graph is adjacent or not. Let $\mathcal{G}$ be the set of finite simple graphs and let $\mathcal{G}(V)$ be the set of such graphs with vertex set V. We shall consider subsets $\mathcal{P}$ of $\mathcal{G}$ that are closed under isomorphism, which we call *graph properties*. To avoid technicalities, we restrict ourselves to graph properties $\mathcal{P}$ such that $\mathcal{P} \cap \mathcal{G}(V) \neq \emptyset$ whenever $V \neq \emptyset$. For instance, this includes all nontrivial *monotone* and *hereditary* graph properties, which are graph properties that are inherited by subgraphs and by induced subgraphs, respectively. Here, we will focus on monotone properties. The prototypical example of a monotone property is $\mathit{Forb}(F)$, the class of all graphs that do not contain a fixed graph F as a subgraph. More generally, if $\mathcal{P}$ is a monotone property and $\mathcal{F}$ contains all minimal graphs that are not in $\mathcal{P}$, then the graphs that lie in $\mathcal{P}$ are precisely those that do not contain an element of $\mathcal{F}$ as a subgraph. This class of graphs will be denoted by $\mathcal{P} = \mathit{Forb}(\mathcal{F})$. The elements of $\mathit{Forb}(\mathcal{F})$ are said to be *$\mathcal{F}$-free.*

A graph property $\mathcal{P}$ is said to be *testable* if, for every $\varepsilon > 0$, there exist a positive integer $q_{\mathcal{P}} = q_{\mathcal{P}}(\varepsilon)$, called the *query complexity*, and a randomized algorithm $\mathcal{T}_{\mathcal{P}}$, called a *tester*, which may perform at most $q_{\mathcal{P}}$ queries in the input graph, satisfying the following property. For an n-vertex input graph Γ, the algorithm $\mathcal{T}_{\mathcal{P}}$ distinguishes with probability at least $2/3$ between the cases in which Γ satisfies $\mathcal{P}$ and in which Γ is ε-far from satisfying $\mathcal{P}$, that is, in which no graph obtained from Γ by the addition or removal of at most $\varepsilon n^2/2$ edges satisfies $\mathcal{P}$. This may be stated in terms of graph distances: given two graphs Γ and Γ' on the same vertex set V, we may define the *normalized edit distance* between Γ and Γ' by $d_1(\Gamma, \Gamma') = \frac{2}{|V|^2} |E(\Gamma) \triangle E(\Gamma')|$, where $E(\Gamma) \triangle E(\Gamma')$ denotes the symmetric difference of their edge sets. If $\mathcal{P}$ is a graph property, we let the distance between a graph Γ and $\mathcal{P}$ be

$$d_1(\Gamma, \mathcal{P}) = \min\{d_1(\Gamma, \Gamma') \colon V(\Gamma') = V(\Gamma) \text{ and } \Gamma' \in \mathcal{P}\}.$$

For instance, if $\Gamma = K_n$ and $\mathcal{P} = \mathit{Forb}(K_3)$, Turán's Theorem ensures that $\binom{n}{2} - \lfloor n^2/4 \rfloor$ edges need to be removed to produce a graph that is K_3-free. In particular, $d_1(K_n, \mathit{Forb}(K_3)) \to 1/2$. Thus a graph property is testable if there is a tester with bounded query complexity that distinguishes with probability at least $2/3$ between the cases $d_1(\Gamma, \mathcal{P}) = 0$ and $d_1(\Gamma, \mathcal{P}) > \varepsilon$.

The systematic study of property testing was initiated by Goldreich, Goldwasser and Ron [19], and there is a very rich literature on this topic. For instance, regarding testers, Goldreich and Trevisan [20] showed that it is sufficient to consider simpler canonical testers, namely those that randomly choose a subset X of vertices in Γ and then verify whether the induced subgraph $\Gamma[X]$ satisfies some related property $\mathcal{P}'$. For example, if the property being tested is having edge density $1/2$, then the algorithm will choose a random subset X of appropriate size and check whether the edge density of $\Gamma[X]$ is within, say, $\varepsilon/2$ of $1/2$. Regarding testable properties, Alon and Shapira [5] proved that every monotone graph property is testable, and, more generally, that the same holds for hereditary graph properties [4]. For more information about property testing, we refer the reader to [18] and the references therein.

In a similar vein, a function $z \colon \mathcal{G} \to \mathbb{R}$ from the set $\mathcal{G}$ of finite graphs into the real numbers is called a *graph parameter* if it is invariant under relabeling of vertices. A graph parameter $z \colon \mathcal{G} \to \mathbb{R}$ is *estimable* if for every $\varepsilon > 0$ and every large enough graph Γ, the value of $z(\Gamma)$ can be approximated up to an additive error of ε by an algorithm that only has access to a subgraph of Γ induced by a set of vertices of size $q_z = q_z(\varepsilon)$, chosen uniformly at random. The query complexity of such an algorithm is $\binom{q_z}{2}$ and the size q_z is called its *sample complexity*. Estimable parameters have been considered in [14] and were defined in the above level of generality in [9]. They are often called *testable parameters*. Borgs et al. [9, Theorem 6.1] gave a complete characterization of the estimable graph parameters which, in particular, also implies that the distance from monotone graph properties is estimable. Their work uses the concept of graph limits and does not give explicit bounds on the query complexity required for this estimation.

Estimable parameters are closely related with the notion of *tolerant testing*, which was introduced by Parnas, Ron and Rubinfeld [23], and is a generalization of standard property testing. Let $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$. An $(\varepsilon_1, \varepsilon_2)$-tolerant tester for a graph property $\mathcal{P}$ is an algorithm that receives a graph Γ as input and distinguishes between the cases $d_1(\Gamma, \mathcal{P}) \leq \varepsilon_1$ and $d_1(\Gamma, \mathcal{P}) \geq \varepsilon_2$ with probability at least 2/3 and constant query complexity. Fischer and Newman [14] proved that every testable graph property $\mathcal{P}$ has a $(d - \varepsilon, d)$-tolerant tester, for every $d, \varepsilon > 0$. The distance from a graph to $\mathcal{P}$ can then be estimated by successively running such tolerant testers. Since every monotone graph property is testable, it follows that the distance to such a property is estimable. Later, Alon, Shapira and Sudakov [6, Theorem 1.2] proved that the distance to every monotone graph property $\mathcal{P}$ is estimable using a more natural algorithm, which simply computes the distance from the induced sampled graph to $\mathcal{P}$. However, one disadvantage of these approaches is that their analysis relies heavily on stronger versions of the Szemerédi Regularity Lemma [24, 2]. Therefore, their algorithms to estimate the distance from monotone graph properties have a query complexity of order at least $\texttt{TOWER}(\texttt{poly}(1/\varepsilon))$, by which we mean a tower of twos of height that is polynomial in $1/\varepsilon$. Moreover, it follows from a result of Gowers [21] that any approach based on the Szemerédi Regularity Lemma cannot lead to a bound that is better than $\texttt{TOWER}(\texttt{poly}(1/\varepsilon))$.

In this paper, we introduce the concept of *recoverable* graph properties (Definition 10). Roughly speaking, given a function $f \colon (0,1] \to \mathbb{R}$, we say a graph property $\mathcal{P}$ is f-recoverable if every large graph $G \in \mathcal{P}$ is ε-close to admitting a partition $\mathcal{V}$ of its vertex set into at most $f(\varepsilon)$ classes that witnesses pertinence in $\mathcal{P}$ (i.e., such that any graph that can be partitioned in the same way must be in $\mathcal{P}$). We prove the following result for recoverable properties.

▶ **Theorem 1.** *Let $\mathcal{P}$ be an f-recoverable graph property, for some function $f \colon (0,1] \to \mathbb{R}$. Then, for all $\varepsilon > 0$ there is n_0 such that, for any graph Γ with $|V(\Gamma)| \geq n_0$, the graph parameter*

$$z(\Gamma) = d_1(\Gamma, \mathcal{P})$$

can be estimated within an additive error of ε with sample complexity $2^{\texttt{poly}(f(\varepsilon/6)/\varepsilon)}$.

We also show (Theorem 16) that every monotone graph property $\mathit{Forb}(\mathcal{F})$ is f-recoverable for some function f that depends only on the bounds for the weighted graph Removal Lemma (Lemma 12) for the family $\mathcal{F}$ – the Removal Lemma states that if a graph is far from being $\mathcal{F}$-free, then it must contain many copies of some element of $\mathcal{F}$ of bounded size. Thus, our approach can improve the required sample complexity for estimating $d_1(\cdot, \mathit{Forb}(\mathcal{F}))$ for families $\mathcal{F}$ for which there are better bounds for the Removal Lemma. In particular, as a consequence of Theorem 1, Theorem 16 and recent improvements by Fox [15] on the bounds for the Removal Lemma, we have the following result.

▶ **Corollary 2.** *Let $\mathcal{F}$ be a* finite *family of graphs. Then, for all $\varepsilon > 0$ there is n_0 such that, for any graph Γ with $|V(\Gamma)| \geq n_0$, the graph parameter*

$$z(\Gamma) = d_1(\Gamma, \mathit{Forb}(\mathcal{F}))$$

can be estimated within an additive error of ε with sample complexity $\texttt{TOWER}(\texttt{poly}(\log(1/\varepsilon)))$.

We obtain similar results for another bounded graph parameter, which, for a graph family $\mathcal{F}$, counts the number of $\mathcal{F}$-free subgraphs of the input graph Γ. Formally, given a graph $\Gamma \in \mathcal{G}$ and a family $\mathcal{F}$ of graphs, we denote the set of all $\mathcal{F}$-free subgraphs of Γ by $Forb(\Gamma, \mathcal{F}) = \{G \in Forb(\mathcal{F}) : G \text{ is a subgraph of } \Gamma\}$, and we consider the parameter

$$z(\Gamma) = \frac{1}{|V(\Gamma)|^2} \log_2 |\, Forb(\Gamma, \mathcal{F})|. \tag{1}$$

For example, if $\mathcal{F} = \{K_3\}$ and $\Gamma = K_n$, computing z requires estimating the number of K_3-free subgraphs of K_n, which was done by Erdős, Kleitman and Rothschild for $\mathcal{F} = \{K_k\}$ [13] (see also Erdős, Frankl and Rödl [12] for F-free subgraphs):

$$z(K_n) = \frac{1}{n^2} \log_2 |\, Forb(\Gamma, \mathcal{F})| = \frac{1}{n^2} \log_2 2^{\frac{1}{2}\binom{n}{2} + o(n^2)} \to \frac{1}{4}.$$

Counting problems of this type were considered by several people. (See, for instance, the logarithmic density in Bollobás [8].)

▶ **Theorem 3.** *Let Forb($\mathcal{F}$) be an f-recoverable graph property, for some function $f\colon (0,1] \to \mathbb{R}$. Then, for all $\varepsilon > 0$ there is n_0 such that, for any graph Γ with $|V(\Gamma)| \geq n_0$, the graph parameter z defined in* (1) *can be estimated within an additive error of ε with sample complexity $2^{\mathtt{poly}(f(\varepsilon/6)/\varepsilon)}$.*

▶ **Corollary 4.** *Let $\mathcal{F}$ be a* finite *family of graphs. Then, for all $\varepsilon > 0$ there is n_0 such that, for any graph Γ with $|V(\Gamma)| \geq n_0$, the graph parameter z defined in* (1) *can be estimated within an additive error of ε with sample complexity $\mathtt{TOWER}(\mathtt{poly}(\log(1/\varepsilon)))$.*

We should mention that the statement of Theorem 3 does not hold for arbitrary non-monotone properties $\mathcal{P}$. For instance, if $\mathcal{P}$ is the hereditary property of graphs having no independent sets of size three, then K_n and $K_n - E(K_3)$ have quite a different number of subgraphs satisfying $\mathcal{P}$, although their distance is negligible. It follows from [9, Theorem 6.1] that this parameter is not estimable.

The remainder of the paper is structured as follows. In Section 2, we provide preliminary definitions that lead to the concept of a recoverable graph property, which is used to prove Theorems 1 and 3. Indeed, these two theorems are consequences of Theorem 18 and Theorem 19, respectively, which are stated in Section 3.

2 Recoverability

The main objective of this section is to introduce the concept of ε-recoverability and to restate our main results in terms of it.

2.1 Estimation over cluster graphs

A *weighted graph* R over a (finite) set of vertices V is a symmetric function from $V \times V$ to $[0,1]$. A weighted graph R may be viewed as a complete graph (with loops) in which a weight $R(i,j)$ is given to each edge $(i,j) \in V(R) \times V(R)$, where $V(R)$ denotes the vertex set of R. The set of all weighted graphs with vertex set V is denoted by $\mathcal{G}^*(V)$ and we define $\mathcal{G}^*$ as the union of all $\mathcal{G}^*(V)$ for V finite. In particular, a graph G is a rational weighted graph such that $G(i,i) = 0$, for every $i \in V(G)$, and either $G(i,j) = 1$ or $G(i,j) = 0$ for every $(i,j) \in V(G) \times V(G)$, $i \neq j$. For a weighted graph $R \in \mathcal{G}^*(V)$ and for sets $A, B \subset V$, we denote $e_R(A,B) = \sum_{(i,j) \in A \times B} R(i,j)$ and $e(R) = e(V,V)$.

Let $k > 0$ and let $R \in \mathcal{G}^*(V)$ be a weighted graph. We define a weighted graph $\mathbb{G}(k, R) \in \mathcal{G}^*([k])$ by assigning weight $R(x_i, x_j)$ to each edge $(i, j) \in [k] \times [k]$, where $\{x_i\}_{i=1}^k$ is a multiset of k vertices of V such that each x_i is chosen with uniform probability, independently of the others (with repetition). With this, we may define estimable parameters in the context of weighted graphs. Henceforth we write $b = a \pm x$ for $a - x \leq b \leq a + x$.

▶ **Definition 5.** We say that a function $z\colon \mathcal{G}^* \to \mathbb{R}$ (also called a *weighted graph parameter*) is *estimable* with *sample complexity* $q\colon (0, 1) \to \mathbb{N}$ if, for every $\varepsilon > 0$ and every weighted graph $\Gamma^* \in \mathcal{G}^*(V)$ with $|V| \geq q(\varepsilon)$, we have $z(\Gamma^*) = z(\mathbb{G}(q, \Gamma^*)) \pm \varepsilon$ with probability at least $2/3$.

Given a graph G and vertex sets $U, W \subseteq V(G)$, let $E_G(U, W) = \{(u, w) \in V(G) \times V(G) : u \in U, w \in W\}$ and $e_G(U, W) = |E_G(U, W)|$. An *equipartition* $\mathcal{V} = \{V_i\}_{i=1}^k$ of a weighted graph R is a partition of its vertex set $V(R)$, such that $|V_i| \leq |V_j| + 1$ for all $(i, j) \in [k] \times [k]$. We often abuse terminology and say that $\mathcal{V}$ is a partition of R.

Let $\mathcal{V} = \{V_1, \ldots, V_k\}$ be an equipartition of a graph G. The *cluster graph* of G by $\mathcal{V}$ is a weighted graph $G/_{\mathcal{V}} \in \mathcal{G}^*([k])$ such that $G/_{\mathcal{V}}\,(i, j) = e_G(V_i, V_j)/(|V_i||V_j|)$ for all $(i, j) \in [k] \times [k]$. For a fixed integer $K > 0$, the set of all equipartitions of a vertex set V into at most K classes will be denoted by $\Pi_K(V)$. We also define the set $G/_{\Pi_K} = \{G/_{\mathcal{V}} : \mathcal{V} \in \Pi_K(V(G))\}$ of all cluster graphs of G of vertex size at most K. The following result states that graph parameters that can be expressed as the optimal value of some optimization problem over $G/_{\Pi_K}$ can be estimated with a query complexity that is only exponential in K and in the error parameter.

▶ **Theorem 6.** *Let $z\colon \mathcal{G} \to \mathbb{R}$ be a graph parameter and suppose that there is a weighted graph parameter $z^*\colon \mathcal{G}^* \to \mathbb{R}$ and constants $K > 0$ and $c > 0$ such that:*
1. *$z(\Gamma) = \min_{R \in \Gamma/_{\Pi_K}} z^*(R)$, for every $\Gamma \in \mathcal{G}$ and*
2. *$|z^*(R) - z^*(R')| \leq c \cdot d_1(R, R')$, for all weighted graphs $R, R' \in \mathcal{G}^*$ on the same vertex set.*

Then z is estimable with sample complexity $\varepsilon \mapsto 2^{poly(K, c/\varepsilon)}$.

The proof of Theorem 6 is based on the following lemma, which asserts that the set of cluster graphs of a graph Γ is very 'similar' to the set of cluster graphs of 'large enough' samples of Γ.

▶ **Lemma 7.** *Given $K > 0$, $\varepsilon > 0$ there is $q = 2^{poly(K, 1/\varepsilon)}$ and n_0 such that the following holds. Consider a graph Γ on $n \geq n_0$ vertices and a random sample $\overline{\Gamma} = \mathbb{G}(q, \Gamma)$ with vertex sets V and $\overline{V}$, respectively. Then, with probability at least $2/3$, we have*
1. *for each $\mathcal{V} \in \Pi_K(V)$, there is a $\overline{\mathcal{V}} \in \Pi_K(\overline{V})$ with $d_1(\Gamma/_{\mathcal{V}}, \overline{\Gamma}/_{\overline{\mathcal{V}}}) \leq \varepsilon$;*
2. *for each $\overline{\mathcal{V}} \in \Pi_K(\overline{V})$, there is a $\mathcal{V} \in \Pi_K(V)$ with $d_1(\Gamma/_{\mathcal{V}}, \overline{\Gamma}/_{\overline{\mathcal{V}}}) \leq \varepsilon$.*

We now deduce Theorem 6 from Lemma 7.

Proof of Theorem 6. Fix $\varepsilon > 0$ and an input graph $\Gamma \in \mathcal{G}(V)$. Let q be as in Lemma 7 with input K and ε/c. We will show that if $\overline{\Gamma} = \mathbb{G}(q, \Gamma)$, then $z(\Gamma) = z(\overline{\Gamma}) \pm \varepsilon$ with probability at least $2/3$.

Let $\mathcal{V} \in \Pi_K(V)$ be an equipartition of Γ such that $z(\Gamma) = z^*(\Gamma/_{\mathcal{V}})$. By Lemma 7, with probability at least $2/3$, there is a partition $\overline{\mathcal{V}}$ of $\overline{\Gamma}$ such that $d_1(\Gamma/_{\mathcal{V}}, \overline{\Gamma}/_{\overline{\mathcal{V}}}) < \varepsilon/c$. By the second condition on z^* in the statement of Theorem 6, we have $|z^*(\overline{\Gamma}/_{\overline{\mathcal{V}}}) - z^*(\Gamma/_{\mathcal{V}})| \leq \varepsilon$, and therefore $z(\overline{\Gamma}) \leq z^*(\overline{\Gamma}/_{\overline{\mathcal{V}}}) \leq z^*(\Gamma/_{\mathcal{V}}) + \varepsilon = z(\Gamma) + \varepsilon$.

A symmetric argument shows that $z(\Gamma) \leq z(\overline{\Gamma}) + \varepsilon$. ◀

In Section 3 we show how to express the parameters we are interested in, namely, $d_1(\Gamma, \mathit{Forb}(\mathcal{F}))$ and $|\,\mathit{Forb}(\Gamma, \mathcal{F})|$, as solutions of suitable optimization problems over the set Γ/Π_K of cluster graphs of Γ.

2.2 Recovering partitions

The *distance* between two weighted graphs $R, R' \in \mathcal{G}^*(V)$ on the same vertex set V is given by

$$d_1(R, R') = \frac{1}{|V|^2} \sum_{(i,j) \in V \times V} |R(i,j) - R'(i,j)|.$$

Let $\mathcal{H} \subseteq \mathcal{G}^*$ be a property of weighted graphs, i.e., a subset of weighted graphs which is closed under isomorphisms. We define

$$d_1(R, \mathcal{H}) = \min_{\substack{R' \in \mathcal{H}: \\ V(R') = V(R)}} d_1(R, R').$$

We assume that $\mathcal{H}$ contains weighted graphs with vertex sets of all possible sizes.

We are interested in the property of graphs that are free of *copies* of members of a (possibly infinite) family $\mathcal{F}$ of graphs. To relate this property to a property of cluster graphs, we introduce some preliminary definitions. Let $\varphi\colon V(F) \to V(R)$ be a mapping from the set of vertices of a graph $F \in \mathcal{G}$ to the set of vertices of a weighted graph $R \in \mathcal{G}^*$. The *homomorphism weight* $\mathit{hom}_\varphi(F, R)$ of φ is defined as

$$\mathit{hom}_\varphi(F, R) = \prod_{(i,j) \in E(F)} R(\varphi(i), \varphi(j)).$$

The *homomorphism density* $t(F, R)$ of $F \in \mathcal{G}$ in $R \in \mathcal{G}^*$ is defined as the average homomorphism weight of a mapping in $\Phi := \{\varphi\colon V(F) \to V(R)\}$, that is,

$$t(F, R) = \frac{1}{|\Phi|} \sum_{\varphi \in \Phi} \mathit{hom}_\varphi(F, R).$$

Note that, if F and R are graphs, then $t(F, R)$ is roughly the subgraph density of F in R (and converges to this quantity when the size of R tends to infinity). Since weighted graphs will represent cluster graphs associated with a partition of the vertex set of the input graph, it will be convenient to work with the following property of weighted graphs:

$$\mathit{Forb}^*_{\mathit{hom}}(\mathcal{F}) = \{R \in \mathcal{G}^* : t(F, R) = 0 \text{ for every } F \in \mathcal{F}\}.$$

Let $R, S \in \mathcal{G}^*(V)$ be weighted graphs on the same set V of vertices. We say that S is a *subgraph* of R, which will be denoted by $S \leq R$, if $S(i,j) \leq R(i,j)$ for every $(i,j) \in V \times V$. Moreover, for a subset $Q \subseteq V$, let $R[Q]$ denote the induced weighted subgraph of R with vertex set Q. We also define $\mathit{Forb}^*_{\mathit{hom}}(R, \mathcal{F}) = \{S \in \mathit{Forb}^*_{\mathit{hom}}(\mathcal{F}) : S \leq R\}$.

The following result shows that having a cluster graph in $\mathit{Forb}^*_{\mathit{hom}}(\mathcal{F})$ witnesses pertinence in $\mathit{Forb}(\mathcal{F})$.

▶ **Proposition 8.** *Let $\mathcal{F}$ be a family of graphs and let $\mathcal{V}$ be an equipartition of a graph G. If $G/\mathcal{V} \in \mathit{Forb}^*_{\mathit{hom}}(\mathcal{F})$, then $G \in \mathit{Forb}(\mathcal{F})$.*

Proof. Let $\mathcal{V} = \{V_i\}_{i=1}^k$ be an equipartition of G and let $R = G/_{\mathcal{V}}$. Fix an arbitrary element $F \in \mathcal{F}$ and an arbitrary injective mapping $\varphi\colon V(F) \hookrightarrow V(G)$. Define the function $\psi\colon V(F) \to V(R)$ by $\psi(v) = i$ if $\varphi(v) \in V_i$. Now, if $t(F,R) = 0$, there must be some edge $(u,w) \in E(F)$ such that $R(\psi(u),\psi(w)) = 0$, which implies that $G(\varphi(u),\varphi(v)) = 0$. Hence, $hom_\varphi(F,G) = 0$. Since φ and F were taken arbitrarily, we must have $G \in Forb(\mathcal{F})$. ◀

It is easy to see that the converse of Proposition 8 does not hold in general. Indeed, there exist graph families $\mathcal{F}$ and graphs $G \in Forb(\mathcal{F})$ such that $G/_{\mathcal{V}}$ is actually very *far* from being in $Forb^*_{hom}(\mathcal{F})$ for some equipartition $\mathcal{V}$ of G. For one such example, let G be the n-vertex bipartite Turán graph $T_2(n)$ for K_3 with partition $V(G) = A \cup B$ and consider $\mathcal{V} = \{V_i\}_{i=1}^t$ with $V_i = A_i \cup B_i$, $i = 1,\dots,t$, where $\{A_i\}_{i=1}^t$ and $\{B_i\}_{i=1}^t$ are equipartitions of A and B respectively. Then $G/_{\mathcal{V}}$ is a complete graph with weight $1/2$ on every edge, so that it is $1/4$-far from being in $Forb^*_{hom}(\{K_3\})$ by Turán's Theorem. More generally, if $\mathcal{V}$ is a random equitable partition of a triangle-free graph $G \in Forb(\{K_3\})$ with large edge density, then with high probability the cluster graph $G/_{\mathcal{V}}$ is still $1/4$-far from being in $Forb^*_{hom}(\{K_3\})$.

On the other hand, we will prove that there exist partitions for graphs in $Forb(\mathcal{F})$ with respect to which an approximate version of the converse of Proposition 8 does hold, that is, we will prove that every graph in $Forb(\mathcal{F})$ is not too far from having a partition of bounded size that witnesses pertinence in $Forb(\mathcal{F})$. We say that such a partition is *recovering* with respect to $Forb(\mathcal{F})$. In what follows, we define recovering partitions formally and in a more general setting.

For every weighted graph $S \in \mathcal{G}^*$, let $\mathcal{G}_S \subseteq \mathcal{G}$ be the graph property of being *reducible* to S, that is,

$$\mathcal{G}_S = \{G \in \mathcal{G} : S = G/_{\mathcal{V}} \text{ for some equipartition } \mathcal{V} \text{ of } G\}.$$

Moreover, let $\mathcal{P}^*$ be the weighted graph property consisting of all cluster graphs that witness pertinence in $\mathcal{P}$, i.e., $\mathcal{P}^* = \{S \in \mathcal{G}^* \colon \emptyset \neq \mathcal{G}_S \subseteq \mathcal{P}\}$. The following observation motivates this definition: if $S \in \mathcal{P}^*$, then verifying that $G \in \mathcal{G}_S$ is a way of determining that $G \in \mathcal{P}$. As a consequence, if we could find a size $K = K(\mathcal{P})$ such that every $G \in \mathcal{P}$ has an equipartition $\mathcal{V}$ of size at most K such that $G/_{\mathcal{V}} \in \mathcal{P}^*$, then we would be able to decide whether $G \in \mathcal{P}$ by simply testing whether it is reducible to some $S \in \mathcal{P}^*$ of order at most K. Also note that, in the case of monotone properties $\mathcal{P} = Forb(\mathcal{F})$, we have $\mathcal{P}^* = Forb^*_{hom}(\mathcal{F})$.

▶ **Definition 9.** An equipartition $\mathcal{V}$ of a graph $G \in \mathcal{P}$ is ε-*recovering* for $\mathcal{P}$ if

$$d_1(G/_{\mathcal{V}}, \mathcal{P}^*) \leq \varepsilon.$$

For monotone properties, this means that an equipartition $\mathcal{V}$ of a graph $G \in Forb(\mathcal{F})$ is ε-recovering for $Forb(\mathcal{F})$ if $d_1(G/_{\mathcal{V}}, Forb^*_{hom}(\mathcal{F})) \leq \varepsilon$, which is the approximate converse of Proposition 8 mentioned above. With this, we say that a graph property $\mathcal{P}$ is recoverable if, for every $\varepsilon > 0$, large graphs satisfying $\mathcal{P}$ admit a constant size ε-recovering partition for $\mathcal{P}$.

▶ **Definition 10.** Let $\mathcal{P}$ be a graph property. For a fixed function $f\colon (0,1] \to \mathbb{R}$, we say that the class $\mathcal{P}$ is f-*recoverable* if, for every $\varepsilon > 0$, there exists $n_0 = n_0(\varepsilon)$ such that the following holds. For every graph $G \in \mathcal{P}$ on $n \geq n_0$ vertices, there is an equipartition $\mathcal{V}$ of G of size $|\mathcal{V}| \leq f(\varepsilon)$ which is ε-recovering for $\mathcal{P}$.

As a simple example, one can verify that the graph property $\mathcal{P}$ of being r-colorable is f-recoverable for $f(\varepsilon) = r/\varepsilon$; here and in what follows, for simplicity, we ignore divisibility conditions and drop floor and ceiling signs. Let G be a graph in $\mathcal{P}$, with color classes $C_1, \ldots, C_r$. Let $k = r/\varepsilon$. Start by fixing parts $V_1, \ldots, V_t$ of size n/k each, with each V_i contained in some C_j $(j = j(i))$, and leaving out fewer than n/k vertices from each C_j $(1 \leq j \leq r)$. The sets V_i $(1 \leq i \leq t)$ cover a subset C_j' of C_j and $X_j = C_j \setminus C_j'$ is left over. We then complete the partition by taking arbitrary parts $U_1, \ldots, U_{k-t}$ of size n/k each, forming a partition of $\bigcup_{1 \leq j \leq r} X_j$. The cluster graph $G/\mathcal{V}$ can be made r-partite by giving weight zero to every edge incident to vertices corresponding to $U_1, \ldots, U_{k-t}$. Therefore $G/\mathcal{V}$ is at distance at most $r/k \leq \varepsilon$ from being r-partite. But since every r-partite weighted graph S clearly satisfies $\mathcal{G}_S \subseteq \mathcal{P}$, we get that $d_1(G/\mathcal{V}, \mathcal{P}^*) \leq \varepsilon$, as required.

Another interesting easy example is the property of *tournaments* that are *transitive*. A tournament — i.e., a complete graph whose edges are given an orientation — is said to be transitive if it does not contain any cycle or, equivalently, if there is a linear ordering $v_1, \ldots, v_n$ of its vertices such that (v_i, v_j) is an arc for every $i < j$. Computing the distance of a tournament T from being transitive, also called the *Slater index* of T, is an interesting problem which has received some attention in the past (see [10] for a survey) and has applications in many areas like psychometrics and voting theory (cf. [7]). Tournaments do not fit exactly into the framework presented here; it would be necessary to make some minor generalizations. However it is easy to see that, given a tournament T with a linear ordering $v_1, \ldots, v_n$, any equipartition $\mathcal{V} = \{V_i\}_{i=1}^k$ respecting this order (i.e., such that for every $1 \leq i < j \leq k$ and $u \in V_i, v \in V_j$ it holds that (u, v) is an arc) is such that $T/\mathcal{V}$ is a transitive directed graph with a loop on every vertex and, therefore, at distance at most $1/k$ from being a transitive tournament. We conclude that the property of being transitive is f-recoverable, with $f(\varepsilon) = 1/\varepsilon$. By Theorem 1, this is sufficient to show that one can estimate the distance of a tournament from being transitive with sample complexity that is *only exponential* in the error parameter ε. We shall elaborate on this in the full version of this paper.

We end this section by noting that the definition of f-recoverable properties has some similarity with the notion of *regular-reducible* properties $\mathcal{P}$ defined by Alon, Fischer, Newman and Shapira [3]. The main difference is that the notion of being regular-reducible requires that every graph $G \in \mathcal{P}$ should have a *regular* partition such that $G/\mathcal{V}$ is close to some property $\mathcal{R}^*$ of weighted graphs, while the definition of f-recoverable properties does not require the partitions to be regular. Another difference is that $\mathcal{R}^*$ must be such that having a (regular) cluster graph in $\mathcal{R}^*$ witnesses only *proximity* (and not pertinence) to $\mathcal{P}$.

2.3 Monotone graph properties are recoverable

Szemerédi's Regularity Lemma [24] can be used to show that every monotone (and actually every hereditary) graph property is f-recoverable, for $f(\varepsilon) = \texttt{TOWER}(\texttt{poly}(1/\varepsilon))$. In the remainder of this section, we prove that monotone properties $\mathcal{P} = \mathit{Forb}(\mathcal{F})$ are recoverable using a weaker version of regularity along with the Removal Lemma, which leads to an improvement on the growth of f for families $\mathcal{F}$ where the Removal Lemma is known to hold with better bounds than the Regularity Lemma.

The Removal Lemma was first stated explicitly in the literature by Alon *et al.* [1] and by Füredi [17]. The following version, which holds for possibly infinite families of graphs was first proven in [5].

▶ **Lemma 11** (Removal Lemma). *For every $\varepsilon > 0$ and every (possibly infinite) family $\mathcal{F}$ of graphs, there exist $M = M(\varepsilon, \mathcal{F})$, $\delta = \delta(\varepsilon, \mathcal{F}) > 0$ and $n_0 = n_0(\varepsilon, \mathcal{F})$ such that the following holds. If a graph G on $n \geq n_0$ vertices satisfies $d_1(G, \mathit{Forb}(\mathcal{F})) \geq \varepsilon$, then there is $F \in \mathcal{F}$ with $|F| \leq M$ such that $t(F, G) \geq \delta$.*

We derive, from Lemma 11, a slightly stronger version of the Removal Lemma, that deals with weighted graphs and homomorphic copies.

▶ **Lemma 12.** *For every $\varepsilon > 0$ and every (possibly infinite) family $\mathcal{F}$ of graphs, there exist $\delta = \delta(\varepsilon, \mathcal{F})$, $M = M(\varepsilon, \mathcal{F})$ and $n_0 = n_0(\varepsilon, \mathcal{F})$ such that the following holds. If a weighted graph R such that $|V(R)| > n_0$ satisfies $d_1(R, \mathrm{Forb}^*_{hom}(\mathcal{F})) \geq \varepsilon$, then there is a graph $F \in \mathcal{F}$ with $|F| \leq M$ such that $t(F, R) \geq \delta$.*

Next, to introduce the version of regularity that we use in this work, we use a second well-known distance between weighted graphs. Let $R_1, R_2 \in \mathcal{G}^*(V)$ be weighted graphs with $|V| = n$. The *cut-distance* between R_1 and R_2 is defined as

$$d_\square(R_1, R_2) = \frac{1}{n^2} \max_{S,T \subseteq V} |e_{R_1}(S,T) - e_{R_2}(S,T)|.$$

Let $\Gamma \in \mathcal{G}(V)$ and $\mathcal{V} = \{V_i\}_{i=1}^k$ be a partition of V. We define the weighted graph $\Gamma_\mathcal{V} \in \mathcal{G}^*(V)$ as the weighted graph such that $\Gamma_\mathcal{V}(u,v) = \Gamma/\mathcal{V}\,(i,j)$ if $u \in V_i$ and $v \in V_j$. Graph regularity lemmas ensure that, for any large graph Γ, there exists an equitable partition $\mathcal{V}$ of constant size such that $\Gamma_\mathcal{V}$ is a faithful approximation of Γ. Here, we use the regularity introduced by Frieze and Kannan [16].

▶ **Definition 13.** A partition $\mathcal{V} = \{V_i\}_{i=1}^k$ of a graph Γ is *γ-FK-regular* if $d_\square(\Gamma, \Gamma_\mathcal{V}) \leq \gamma$, or, equivalently if for all $S, T \subseteq V(\Gamma)$ it holds that

$$e(S,T) = \sum_{(i,j) \in [k] \times [k]} |S \cap V_i||T \cap V_j|\ \Gamma/\mathcal{V}\,(i,j) \pm \gamma |V(\Gamma)|^2.$$

▶ **Lemma 14** (Frieze-Kannan Regularity Lemma). *For every $\gamma > 0$ and every $t_0 > 0$, there is $T = t_0 \cdot 2^{poly(1/\gamma)}$ such that every graph Γ on $n \geq T$ vertices admits a γ-FK-regular equipartition into t classes, where $t_0 \leq t \leq T$.*

Conlon and Fox [11] found instances where the number t of classes in any γ-FK-regular equipartition is at least $t \geq 2^{1/(2^{60}\gamma^2)}$ (for a previous result, see Lovász and Szegedy [22]).

We will also need the following result, which states that a graph has homomorphism densities close to the ones of the cluster graphs with respect to FK-regular partitions.

▶ **Lemma 15** ([9, Lemma 2.7(a)]). *Let $\mathcal{V}$ be a γ-FK-equipartition of a graph $G \in \mathcal{G}$. Then, for any graph $F \in \mathcal{G}$ it holds that $t(F, G) = t(F, G_\mathcal{V}) \pm 4e(F)\gamma = t(F, G/\mathcal{V}) \pm 4e(F)\gamma$.*

We are now ready to show that every monotone graph property is f-recoverable.

▶ **Theorem 16.** *For every family $\mathcal{F}$ of graphs, the property $\mathrm{Forb}(\mathcal{F})$ is f-recoverable for $f(\varepsilon) = n_0 2^{poly(1/\delta, M)}$, where δ, M and n_0 are as in Lemma 12 with input $\mathcal{F}$ and ε.*

Proof. Let δ, M and n_0 be as in Lemma 12 with input $\mathcal{F}$ and ε and let $\gamma = \delta/(3M)^2$. By Lemma 14, it suffices to show that any γ-FK-regular partition $\mathcal{V} = \{V_i\}_{i=1}^k$ of a graph $G \in Forb(\mathcal{F})$ into $k \geq n_0$ classes is ε-recovering.

Let $R = G/\mathcal{V}$ and suppose by contradiction that $d_1(R, \mathrm{Forb}^*_{hom}(\mathcal{F})) \geq \varepsilon$. Then, by Lemma 12, we have $t(F, R) \geq \delta$ for some graph $F \in \mathcal{F}$ such that $|F| \leq M$. By Lemma 15, we would have $t(F, G) \geq \delta - 2\gamma M^2 > 0$, a contradiction to $G \in Forb(\mathcal{F})$. ◀

3 Estimation of $d_1(\Gamma, \mathcal{F})$ and $|\,Forb(\Gamma, \mathcal{F})|$

The objective of this section is to prove Theorems 1 and 3. For that, we shall use the following fact about equipartitions, whose simple proof is omitted.

▶ **Lemma 17.** *Let $\Gamma, G \in \mathcal{G}(V)$ for some vertex set V and let $\mathcal{V}$ be any equipartition of V. Then $d_1(\Gamma/_{\mathcal{V}}, G/_{\mathcal{V}}) \le d_1(\Gamma, G) + |\mathcal{V}|/|V|$.*

The final ingredient needed for Theorem 1 is the result below, which, for a recoverable property $\mathcal{P}$, relates the parameter $d_1(\cdot, \mathcal{P})$ with a parameter to which Theorem 6 may be applied.

▶ **Theorem 18.** *Let $\mathcal{P}$ be an f-recoverable graph property for some function $f\colon (0,1] \to \mathbb{R}$. Fix $\varepsilon > 0$ and let $K = f(\varepsilon/2)$. Then every graph $\Gamma \in \mathcal{G}(V)$ such that $|V| > 2K/\varepsilon$ satisfies*

$$d_1(\Gamma, \mathcal{P}) = \min_{R \in \Gamma/_{\Pi_K}} d_1(R, \mathcal{P}^*) \pm \varepsilon.$$

Proof. Fix $0 < \varepsilon < 1$, $K = f(\varepsilon/2)$. Let $V = [n]$ and let $d = d_1(\Gamma, \mathcal{P})$ and $\widehat{d} = \min_{R \in \Gamma/_{\Pi_K}} d_1(R, \mathcal{P}^*)$.

We first show that $\widehat{d} \le d + \varepsilon$. Let $G \in \mathcal{P}$ be a graph such that $d_1(\Gamma, G) = d$. Since $\mathcal{P}$ is f-recoverable, we can fix an $\varepsilon/2$-recovering equipartition $\mathcal{V}$ of size $1 \le k \le K$ of G, i.e., an equipartition satisfying

$$d_1(G/_{\mathcal{V}}, \mathcal{P}^*) \le \frac{\varepsilon}{2}.$$

By Lemma 17 we have

$$d_1(\Gamma/_{\mathcal{V}}, G/_{\mathcal{V}}) \le d_1(\Gamma, G) + \frac{k}{n} \le d + \frac{\varepsilon}{2}.$$

Now we add the last two inequalities and apply the triangle inequality to obtain

$$d + \varepsilon \ge d_1(\Gamma/_{\mathcal{V}}, \mathcal{P}^*) \ge \widehat{d}.$$

Next, we proceed to show that $d \le \widehat{d} + \varepsilon$. Let $R \in \Gamma/_{\Pi_K}$ and $S \in \mathcal{P}^*$ be such that $d_1(R, S) = \widehat{d}$. Let $k = |V(R)|$ and fix an equipartition $\mathcal{V} = \{V_1, \ldots, V_k\}$ of Γ such that $R = \Gamma/_{\mathcal{V}}$. Consider a graph G with vertex set $V(\Gamma)$ such that $G/_{\mathcal{V}} = S$, obtained as follows. For each $(i,j) \in [n] \times [n]$ such that $R(i,j) > S(i,j)$, we remove exactly $(R(i,j) - S(i,j))|V_i||V_j|$ edges from Γ between V_i and V_j; if $S(i,j) > R(i,j)$, we add exactly $(R(i,j) - S(i,j))|V_i||V_j|$ between V_i and V_j to Γ, thus

$$\begin{aligned} d_1(\Gamma, G) &= \frac{1}{n^2} \sum_{(i,j) \in [k] \times [k]} |E_\Gamma(V_i, V_j) \triangle E_G(V_i, V_j)| \\ &= \frac{1}{n^2} \sum_{(i,j) \in [k] \times [k]} |S(i,j) - R(i,j)||V_i||V_j| \\ &\le \frac{1}{n^2} \sum_{(i,j) \in [k] \times [k]} |S(i,j) - R(i,j)| \frac{(n+k)}{k} \frac{(n+k)}{k} \\ &\le \widehat{d} + \frac{k}{n} + \frac{k^2}{n^2} \le \widehat{d} + \varepsilon. \qquad \text{(as } n > 2K/\varepsilon) \end{aligned}$$

Since, by construction, G is reducible to $S \in \mathcal{P}^*$, we must have $G \in \mathcal{P}$. Hence, $d \le d_1(\Gamma, G) \le \widehat{d} + \varepsilon$. ◀

Proof of Theorem 1. Let $\mathcal{P}$ be an f-recoverable graph property. Fix $\varepsilon > 0$ and let $K = f(\varepsilon/6)$, so that by Theorem 18 we have

$$\left| d_1(\Gamma, \mathcal{P}) - \min_{R \in \Gamma/\Pi_K} d_1(R, \mathcal{P}^*) \right| \leq \frac{\varepsilon}{3}, \tag{2}$$

whenever $|V(\Gamma)| > 12K/\varepsilon$.

Let $\widehat{z}\colon \mathcal{G} \to \mathbb{R}$ be the graph parameter defined by $\widehat{z}(\Gamma) = \min_{R\in\Gamma/\Pi_K} z^*(R)$, where $z^*(R) = d_1(R, \mathcal{P}^*)$. By the triangle inequality, given R and R' in $\mathcal{G}^*(V)$, we have $z^*(R) \leq d_1(R, R') + z^*(R')$ and $z^*(R') \leq d_1(R, R') + z^*(R)$, so that $|z^*(R) - z^*(R')| \leq d_1(R, R')$. Theorem 6 applies, and $\widehat{z}$ is estimable with sample complexity $q(\varepsilon) = 2^{\texttt{poly}(K/\varepsilon)}$. Hence, with probability at least 2/3, a sample $\overline{\Gamma} = \mathbb{G}(q(\varepsilon/3), \Gamma)$ of Γ is such that $|\widehat{z}(\overline{\Gamma}) - \widehat{z}(\Gamma)| \leq \varepsilon/3$. By (2) we have $|\, d_1(\Gamma, \mathcal{P}) - \widehat{z}(\Gamma)| \leq \varepsilon/3$. On the other hand, we can also apply (2) to $\overline{\Gamma}$ to obtain $|\widehat{z}(\overline{\Gamma}) - d_1(\overline{\Gamma}, \mathcal{P})| \leq \varepsilon/3$. Using the triangle inequality along with the last three inequalities, we obtain $|\, d_1(\Gamma, \mathcal{P}) - d_1(\overline{\Gamma}, \mathcal{P})| \leq \varepsilon$. ◀

Proof of Corollary 2. Fox [15] proved Lemma 11 when $\mathcal{F} = \{F\}$ avoiding the Szemerédi Regularity Lemma and thus obtained better bounds on the size of $\delta > 0$ (and n_0). More specifically his result implies the following. For every fixed *finite* family $\mathcal{F}$ of graphs, Lemma 11 holds with both $1/\delta$ and n_0 bounded by $\texttt{TOWER}(O(\log(1/\varepsilon)))$ as $M = M(\mathcal{F})$ is a constant. Hence, by Theorem 16 we have that if $\mathcal{F}$ is finite, then *Forb*$(\mathcal{F})$ is f-recoverable, where $f(\varepsilon) = \texttt{TOWER}(\texttt{poly}(\log(1/\varepsilon)))$. ◀

The structure of the proof of Theorem 3 is analogous to that of Theorem 1. Recall that $Forb^*_{hom}(R, \mathcal{F}) = \{S \leq R : t(F, S) = 0, \text{for every } F \in \mathcal{F}\}$, and set

$$ex^*(R, \mathcal{F}) = \frac{1}{2|V(R)|^2} \max_{S\in Forb^*_{hom}(R,\mathcal{F})} e(S),$$

which measures the largest edge density of a subgraph of R not containing a copy of any $F \in \mathcal{F}$.

We shall derive Theorem 3 from the following auxiliary result, whose proof is omitted.

▶ **Theorem 19.** *Let $\mathcal{F}$ be a family of graphs such that* Forb$(\mathcal{F})$ *is f-recoverable for some function $f\colon (0, 1] \to \mathbb{R}$. Then, for any $\varepsilon > 0$, there exists $K = f(\texttt{poly}(1/\varepsilon))$ and $N = \texttt{poly}(K)$ such that for any graph Γ of size $n \geq N$ it holds that*

$$\frac{\log_2 |\, Forb(\Gamma, \mathcal{F})|}{n^2} = \max_{R\in\Gamma/\Pi_K} ex^*(R, \mathcal{F}) \pm \varepsilon.$$

Proof of Theorem 3. Let $\mathcal{F}$ be a family of graphs such that *Forb*$(\mathcal{F})$ is f-recoverable. Fix $\varepsilon > 0$ and let $K = f(\varepsilon/6)$, so that by Theorem 19 we have

$$\left| \frac{\log_2 |\, Forb(\Gamma, \mathcal{F})|}{n^2} - \max_{R\in\Gamma/\Pi_K} ex^*(R, \mathcal{F}) \right| \leq \frac{\varepsilon}{3},$$

whenever $|V(\Gamma)| > N$.

Let $\widehat{z}\colon \mathcal{G} \to \mathbb{R}$ be the graph parameter defined by $\widehat{z}(\Gamma) = \max_{R\in\Gamma/\Pi_K} z^*(R)$, where $z^*(R) = ex^*(R, \mathcal{F})$. We claim that, given R and R' in $\mathcal{G}^*(V)$, we have $|z^*(R) - z^*(R')| \leq d_1(R, R')$. Indeed, assume without loss of generality that $z^*(R) \geq z^*(R')$ and fix a subgraph $S \leq R$ such that $S \in Forb^*_{hom}(R, \mathcal{F})$ and $z^*(R) = e(S)/(2|V(R)|^2)$. If $S \in Forb^*_{hom}(R', \mathcal{F})$, we are done, so assume that this is not the case. Let S' be a subgraph of S and R' maximizing $e(S')$. Clearly,

$$e(S') \geq e(S) - \frac{1}{2} \sum_{(i,j) \in V \times V} |R(i,j) - R'(i,j)| \geq e(S) - |V|^2\, d_1(R, R'),$$

so that $0 \leq z^*(R) - z^*(R') \leq \frac{1}{2}|V|^{-2}\,(e(S) - e(S')) \leq d_1(R, R')$.

We now apply Theorem 6 to conclude that $\widehat{z}$ is estimable with sample complexity $q(\varepsilon) = 2^{\mathtt{poly}(K/\varepsilon)}$. It follows that, with probability at least $2/3$, a sample $\overline{\Gamma} = \mathbb{G}(q(\varepsilon/3), G)$ of G is such that $|\widehat{z}(\overline{\Gamma}) - \widehat{z}(\Gamma)| \leq \varepsilon/3$. By (2) we have $\left|n^{-2} \log_2 |\mathit{Forb}(\Gamma, \mathcal{F})| - \widehat{z}(\Gamma)\right| \leq \varepsilon/3$. On the other hand, we can also apply (2) to $\overline{\Gamma}$ to obtain $\left|\widehat{z}(\overline{\Gamma}) - q(\varepsilon/3)^{-2} \log_2 |\mathit{Forb}(\overline{\Gamma}, \mathcal{F})|\right| \leq \varepsilon/3$. By adding the last three inequalities, we get that

$$\left| \frac{1}{n^2} \log_2 |\mathit{Forb}(\Gamma, \mathcal{F})| - \frac{1}{q(\varepsilon/3)^2} \log_2 |\mathit{Forb}(\overline{\Gamma}, \mathcal{F})| \right| \leq \varepsilon,$$

as required. ◀

Corollary 4 follows directly from Theorem 3, just as Corollary 2 is a direct consequence of Theorem 1.

4 Concluding remarks

Here, we have restricted ourselves to graphs and graph properties. No substantial problems arise if one wishes to cover tournaments or directed graphs: it suffices to consider ordered graphs, that is, graphs whose vertex sets are linearly ordered, with weights on the edges, with negative weights allowed (in fact, one can consider matrices with entries in $[-1, 1]$). Details are worked out in the journal version of this extended abstract.

We believe it would be interesting to investigate in more detail the notion of recoverability. For instance, when is a property $f(\varepsilon)$-recoverable for $f(\varepsilon)$ polynomial in $1/\varepsilon$?

Acknowledgements. The authors thank the referees for their helpful comments.

References

1 Noga Alon, Richard A. Duke, Hanno Lefmann, Vojtěch Rödl, and Raphael Yuster. The algorithmic aspects of the regularity lemma. *J. Algorithms*, 16(1):80–109, 1994. doi:10.1006/jagm.1994.1005.

2 Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20(4):451–476, 2000. doi:10.1007/s004930070001.

3 Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity. *SIAM J. Comput.*, 39(1):143–167, 2009. doi:10.1137/060667177.

4 Noga Alon and Asaf Shapira. A characterization of the (natural) graph properties testable with one-sided error. *SIAM J. Comput.*, 37(6):1703–1727, 2008. doi:10.1137/06064888X.

5 Noga Alon and Asaf Shapira. Every monotone graph property is testable. *SIAM J. Comput.*, 38(2):505–522, 2008. doi:10.1137/050633445.

6 Noga Alon, Asaf Shapira, and Benny Sudakov. Additive approximation for edge-deletion problems. *Ann. of Math. (2)*, 170(1):371–411, 2009. doi:10.4007/annals.2009.170.371.

7 Jean-Pierre Barthélémy and Bernard Monjardet. The median procedure in cluster analysis and social choice theory. *Math. Social Sci.*, 1(3):235–267, 1980/81. doi:10.1016/0165-4896(81)90041-X.

8 Béla Bollobás. Hereditary properties of graphs: asymptotic enumeration, global structure, and colouring. *Doc. Math.*, pages 333–342 (electronic), 1998. Extra Vol. III.

9 Christian Borgs, Jennifer T. Chayes, László Lovász, Vera T. Sós, and Katalin Vesztergombi. Convergent sequences of dense graphs. I. Subgraph frequencies, metric properties and testing. *Adv. Math.*, 219(6):1801–1851, 2008. `doi:10.1016/j.aim.2008.07.008`.

10 Irène Charon and Olivier Hudry. An updated survey on the linear ordering problem for weighted or unweighted tournaments. *Ann. Oper. Res.*, 175:107–158, 2010. `doi:10.1007/s10479-009-0648-7`.

11 David Conlon and Jacob Fox. Bounds for graph regularity and removal lemmas. *Geom. Funct. Anal.*, 22(5):1191–1256, 2012. `doi:10.1007/s00039-012-0171-x`.

12 Paul Erdős, Péter Frankl, and Vojtěch Rödl. The asymptotic number of graphs not containing a fixed subgraph and a problem for hypergraphs having no exponent. *Graphs and Combinatorics*, 2(1):113–121, 1986. `doi:10.1007/BF01788085`.

13 Paul Erdős, Daniel J. Kleitman, and Bruce L. Rothschild. Asymptotic enumeration of K_n-free graphs. In *Colloquio Internazionale sulle Teorie Combinatorie (Rome, 1973), Tomo II*, pages 19–27. Atti dei Convegni Lincei, No. 17. Accad. Naz. Lincei, Rome, 1976.

14 Eldar Fischer and Ilan Newman. Testing versus estimation of graph properties. *SIAM J. Comput.*, 37(2):482–501 (electronic), 2007. `doi:10.1137/060652324`.

15 Jacob Fox. A new proof of the graph removal lemma. *Ann. of Math. (2)*, 174(1):561–579, 2011. `doi:10.4007/annals.2011.174.1.17`.

16 Alan Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999. `doi:10.1007/s004930050052`.

17 Z. Füredi. Extremal hypergraphs and combinatorial geometry. In S. D. Chatterji, editor, *Proceedings of the International Congress of Mathematicians: August 3–11, 1994 Zürich, Switzerland*, pages 1343–1352. Birkhäuser Basel, 1995. `doi:10.1007/978-3-0348-9078-6_65`.

18 Oded Goldreich, editor. *Property Testing – Current Research and Surveys [outgrow of a workshop at the Institute for Computer Science ITCS) at Tsinghua University, January 2010]*, volume 6390 of *Lecture Notes in Computer Science*. Springer, 2010. `doi:10.1007/978-3-642-16367-8`.

19 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`.

20 Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Structures Algorithms*, 23(1):23–57, 2003. `doi:10.1002/rsa.10078`.

21 William T. Gowers. Lower bounds of tower type for Szemerédi's uniformity lemma. *Geom. Funct. Anal.*, 7(2):322–337, 1997. `doi:10.1007/PL00001621`.

22 László Lovász and Balázs Szegedy. Szemerédi's lemma for the analyst. *Geom. Funct. Anal.*, 17(1):252–270, 2007. `doi:10.1007/s00039-007-0599-6`.

23 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. System Sci.*, 72(6):1012–1042, 2006. `doi:10.1016/j.jcss.2006.03.002`.

24 Endre Szemerédi. Regular partitions of graphs. In *Problèmes combinatoires et théorie des graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976)*, volume 260 of *Colloq. Internat. CNRS*, pages 399–401. CNRS, Paris, 1978.

Stable Matching with Evolving Preferences*

Varun Kanade[1], Nikos Leonardos[2], and Frédéric Magniez[3]

1 **University of Oxford, U.K.**
varunk@cs.ox.ac.uk
2 **University of Athens, Greece**
nikos.leonardos@gmail.com
3 **CNRS and IRIF, Univ Paris Diderot, Sorbonne Paris-Cité, France**
frederic.magniez@cnrs.fr

Abstract

We consider the problem of stable matching with dynamic preference lists. At each time-step, the preference list of some player may change by swapping random adjacent members. The goal of a central agency (algorithm) is to maintain an approximately stable matching, in terms of number of blocking pairs, at all time-steps. The changes in the preference lists are not reported to the algorithm, but must instead be probed explicitly. We design an algorithm that in expectation and with high probability maintains a matching that has at most $O((\log n)^2)$ blocking pairs.

1998 ACM Subject Classification F.1.2 Models of Computation

Keywords and phrases Stable Matching, Dynamic Data

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.36

1 Introduction

In the world of massive and distributed data, it is hardly reasonable to assume that data are static. Yet, one must design algorithms that maintain a solution for a given problem that is (approximately) consistent with the requirements, *e.g.,* a permutation that is almost sorted. Thus, it is important to design algorithms and data structures that are robust to changes in their input, *i.e.,* they produce an output with some performance guarantee (quickly).

There are a few different dynamic data models that have been considered. The area of dynamic graph algorithms consists of maintaining some property or structure, such as connectivity, matchings, or spanning trees, even when the underlying graphs are changing [3, 11, 10, 5]. Here, it is assumed that the changes to the graph may be *arbitrary*, but are reported to the algorithm; and the focus is on designing data structures and algorithms that adapt efficiently (typically in terms of computational time) to changes in the input. The area of streaming algorithms studies the setting where the data can only be accessed as a *stream* and the focus is on producing the desired output with highly space-efficient procedures (typically poly-logarithmic in the size of the input). In the area of online algorithms, one must design procedures that, even when data is revealed bit by bit, produce an output that is *competitive* with algorithms that see the entire input at once.

Recently, Anagnostopoulos *et al.* [1] proposed the *evolving data model* to take into account the dynamic aspects of massive data. In this model, the changes to the data are not

* Partially supported by the French ANR Blanc project ANR-12-BS02-005 (RDAM) and the European Commission IST STREP project Quantum Algorithms (QALGO) 600700. Most of the work was carried out when V. K and N. L. were at LIAFA (now IRIF). During this time V.K. was supported by the Fondation Sciences Mathématiques de Paris (FSMP).

revealed to the algorithm; instead, an algorithm has query access to the data. However, it is assumed that the changes to the data are *stochastic*, not adversarial. In this setting, the focus is not on computational complexity (which is allowed to be polynomial at each time-step), but query complexity, the number of probes made by the algorithm. Anagnostopoulos *et al.* [1] studied the problem of maintaining the *minimum* element of a permutation and an approximately sorted permutation, motivated by questions such as maintaining high (page)-ranked pages. In their setup, a permutation evolves by choosing a random element and swapping it with an adjacent element. In later work, Anagnostopoulos *et al.* [2] studied evolving graph models and problems such as s-t connectivity and minimum spanning tree.

In this work, our aim is to bring this notion to game theory starting from the basic problem of computing a stable matching. In other words, we introduce the notion of evolving agents, who may not report any updates to their strategy (or preferences) without an explicit request. In the stable matching problem, the input consists of two sets A, B of equal size, and for each member a total order (preference) over members in the other set. Given a matching between A and B, a pair (x, y) with $x \in A$ and $y \in B$ is blocking if they prefer each other to their matches. A matching is stable if there are no blocking pairs. Gale and Shapley showed that a stable matching always exists and can be found by an efficient algorithm [4]. We consider the setting where the preference lists *evolve* over time. The preference lists can evolve over time, by swapping adjacent elements. More precisely, while the algorithm can perform one query per time-step, we assume that a total number of α swaps events also occur, where $\alpha = \Theta(1)$ is some fixed parameter, called the *evolution rate*. This assumption is similar to previous works and models the critical regime: with less evolution events the input is basically static, and with more the input evolves too fast. The goal is to maintain a matching that has *few* blocking pairs.

We summarize our results as follows. All three statements hold in expectation and with high probability.

1. Using the results of Anagnostopoulos *et al.* [1] for sorting permutations, we design an algorithm that maintains a matching with at most $O(n \log n)$ blocking pairs, at all time-steps after roughly the first $n^2 \log n$ steps (Theorem 6). Also, we observe that any analysis that uses their method as a black-box, cannot improve on this bound (Remark 7).[1]
2. In a restricted setting, where only one side, say the B side, has evolving preference lists, and if the A side has uniform random permutations as preference lists (known to the central agency), we design an algorithm that maintains a matching with $O(\log n)$ blocking pairs at all time-steps after roughly the first $n \log n$ steps (Theorem 8).
3. Finally, we design an algorithm in the general setting, that maintains a matching with at most $O((\log n)^2)$ blocking pairs at all time-steps after roughly the first $n^2 \log n$ steps (Theorem 14).

2 Preliminaries

In the rest of the paper, $n \geq 1$ denotes an integer parameter and $[n]$ the set of integers $\{1, 2, \ldots, n\}$. For a non-negative random variable X, parametrized by some integer n, we write "$X = O(f(n))$ *in expectation and with high probability*" when for any constant c there exist constants $n_0, c', c'' > 0$ such that $\mathbb{E}[X] \leq c' f(n)$ and $\Pr(X > c'' f(n)) \leq n^{-c}$, for every integer $n \geq n_0$.

[1] We don't rule out the possibility that a more fine-grained analysis of the algorithm may give better bounds; instead we design new algorithms.

2.1 Stable Matching

We only consider the bipartite stable matching problem, also known as stable marriage. There are two sets of players A and B, with $|A| = |B| = n$. Each player $x \in A$ ($y \in B$) holds a permutation of B (A), denoted $\pi_x : B \to [n]$ ($\pi_y : A \to [n]$) indicating their preferences over players in the set B (A). Thus, for $y \in B$, $\pi_x(y)$ denotes the rank of y in x's preference list (where 1 is the highest rank).

Let $M : A \to B$ be a matching (a bijection). A pair (x, y) is said to be *blocking* if $y \prec_{\pi_x} M(x)$ and $x \prec_{\pi_y} M^{-1}(y)$, where $z \prec_\pi z'$ indicates that z is ranked higher than z' according to permutation π (*i.e.*, $\pi(z) < \pi(z')$). In words, x prefers y to $M(x)$ and y prefers x to $M^{-1}(y)$.

A matching M is *stable* if there are no blocking pairs. Then the *stable matching problem* is to find a stable matching given preference lists $\{\pi_z : z \in A \cup B)\}$. Gale and Shapley [4] proved that a stable matching always exists, and gave an algorithm that given the preferences lists as input finds a stable matching in $O(n^2)$ time.

The Gale-Shapley algorithm is simple to describe. Only players in the set A make proposals. Initially all players are *unmatched*. Let M denote a partial matching at some point. If there is an unmatched player $x \in A$, x makes a proposal to $y \in B$, where y is the highest-ranked player in π_x to whom x has not yet proposed. If y is unmatched, or prefers x to $M^{-1}(y)$, then y accepts the proposal and we set $M(x) = y$. In the latter case, the agent previously matched to y, *i.e.*, $M^{-1}(y)$ before M was updated, becomes unmatched once more. Gale and Shapley showed that this algorithm always results in a stable matching.

Wilson [12] studied the problem where all the preference lists are independent and uniformly random permutations; in this case, he showed that the number of proposals made by the Gale-Shapley algorithm is $O(n \log n)$ in expectation and with high probability (see also [9]). In fact, only the proposing side needs to be random in their statement. We provide a proof sketch for completeness.

▶ **Theorem 1** ([12]). *If the permutations $\{\pi_x : x \in A\}$ are chosen randomly, the number of proposals made in the Gale-Shapley algorithm (where only A makes proposals) is $O(n \log n)$ in expectation and with high probability.*

Proof Sketch. Following the proof in [9] (see also [6]), analyze an alternative procedure where every proposal is uniform over the whole of B. If it happens that $x \in A$ proposes to a $y \in B$ that has already rejected x, then a rejection is guaranteed. It is not hard to show that the number of proposals such an algorithm makes stochastically dominates the number of proposals of the classical algorithm. Next, by the method of deferred decisions, fix the randomness in the algorithm only when needed. Then observe that the number of proposals is equal to the number of coupons chosen in the coupon-collector's problem. ◀

2.2 Model for evolving input

A general framework for studying dynamic data was introduced in [1]. Here we are only concerned with evolving preference lists (or permutations). In our model, we consider discrete *time-steps*. In each time-step, the algorithm is competing against *nature* as follows: the algorithm can *query* the input locally, *nature* lets the input evolve according to one or more *evolution events*.

A query to the stable matching input is a triplet $(z, u, v) \in (A \times B^2) \cup (B \times A^2)$ and the answer is whether $\pi_z(u) < \pi_z(v)$. One evolution event consists of the following: pick $z \in A \cup B$ and $i \in [n-1]$ uniformly at random and swap $u = \pi_z^{-1}(i)$ and $v = \pi_z^{-1}(i+1)$ (*i.e.*, set $\pi_z(u) = i+1$ and $\pi_z(v) = i$).

While the algorithm can perform one query per time-step, α evolution events also occur, where $\alpha \geq 1$ is some integer called the *evolution rate*. We further assume that $\alpha = \Theta(1)$, meaning that evolution events occur basically as often than the algorithm probes. We emphasize that the rate-limiting factor is the queries made by the algorithm. In particular, the algorithm may perform arbitrary (polynomial-time) computations in between time-steps. We are now ready to define our problem:

Evolving Stable Matching (ESM): Given query access to an instance of the stable matching problem with evolution rate $\alpha = \Theta(1)$, maintain a matching that minimizes the number of blocking pairs.

2.3 Sorting evolving permutations

The problem of sorting a single evolving permutation has been already addressed in [1]. In this context, the evolution rate is still constant, but denotes the evolution rate of this single permutation. We will use the algorithm QUICKSORT of [1]. It is simply the randomized version of quicksort which is shown to be robust with respect to an evolving input. The first lemma shows that the running time of quicksort is not affected by evolution events.[2]

▶ **Lemma 2** (Proposition 3 in [1]). *The running time of* QUICKSORT *is $O(n \log n)$ in expectation and with high probability, for any rate of evolution when the pairs to be swapped are chosen randomly.*

Second, Lemma 6 in [1] states that QUICKSORT when run on an evolving permutation π, computes a permutation $\tilde{\pi}$ in which every element is approximately sorted. At time-step t, let π^t the denote the current permutation, and $\tilde{\pi}^t$ its approximation computed by the algorithm.

▶ **Lemma 3** (Lemma 6 in [1]). *Let t be the time-step of completion of* QUICKSORT, *then given an element u, the number of pairs (u, v) that the permutations π^t and $\tilde{\pi}^t$ rank differently is $O(\log n)$ in expectation and with high probability.*

In our setting there are $2n$ evolving permutations over some set of n elements. Algorithm 1 simply sorts m (out of $2n$) permutations, denoted by $\pi_1, \ldots, \pi_m$ using QUICKSORT one after another. (We always invoke Algorithm 1 with either n or $2n$ permutations.)

Algorithm 1 : Sequential sorting

```
1: procedure SEQUENTIALSORT($\{\pi_j : j = 1, \ldots, m\}$)  ▷ Only have query access to input
2:     for $j = 1$ to $m$ do
3:         $\tilde{\pi}_j \leftarrow$ QUICKSORT($\pi_j$)
4:     return $\{\tilde{\pi}_j : j = 1, \ldots, m\}$
```

Using Lemma 3 (Lemma 6 of [1]) we can argue that Algorithm 1 maintains all permutations approximately sorted. While the evolving rate is still $\alpha = \Theta(1)$, there are now $2n$ evolving permutations, and the total number of evolution events is α per time-step.

[2] We remark that Anagnostopoulos *et al.* [1] use 'whp' to denote events that hold with probability $1 - o(1)$, rather than the stronger notion we use in this paper. However, their proofs for the results used in our paper actually prove the stronger bounds. They have other results that do not satisfy the stronger notion and these are not used in our work.

▶ **Lemma 4.** *Let t be the time-step when Algorithm 1 terminates. Then, for $m \leq 2n$, given any element u and $j \in [m]$, the number of pairs (u, v) that the permutations π_j^t and $\tilde{\pi}_j^t$ rank differently is $O(\log n)$ in expectation and with high probability.*

Proof. Fix some $j \in [n]$. Suppose that $\tilde{\pi}_j^t$ was computed at time-step $t' \leq t$ (the time-step when QUICKSORT for this particular list terminates). By Lemma 3 the statement holds for u at time-step t'. Due to Lemma 2 we have $t - t' = O(n^2 \log n)$ with high probability. During this time, the number of evolution steps that have swapped u with an adjacent element is $O(\log n)$ with high probability. This follows from a balls-and-bins experiment where we throw $O(n^2 \log n)$ balls (corresponding to the evolution steps) into $m(n-1)$ bins (corresponding to the adjacent pairs). It is known (see Exercise 3.1 in [9]) that in this particular case the number of balls in every bin is of the order of its mean with high probability. Therefore, during this time, at most $O(\log n)$ more elements may be swapped with u. ◀

2.4 Chernoff Bound with dependent variables

We will require the following extension of the Chernoff bound. It follows from (the more general) Theorem 3.8 in [8].

▶ **Theorem 5.** *For $i \in [n]$, let Y_i be a random variable over some set $\mathcal{Y}_i$ and X_i be a Boolean random variable. For any $y \in \prod_{i=1}^n \mathcal{Y}_i$ and $k \in [n]$, let $E_k(y)$ denote the event $Y_1 = y_1, \ldots, Y_k = y_k$. Suppose $\mathbb{P}[X_k = 1 | E_{k-1}(y)] \leq p$, for all y and k as above. Then, for any $t \geq 0$,*

$$\mathbb{P}\left[\sum X_k \geq pn + t\right] \leq \exp\left(-\frac{3t^2}{6pn + 2t}\right).$$

3 Two simple cases

In this section we present two simple arguments in two different settings. First, we consider how the original Gale-Shapley algorithm performs when run, without any modification, on lists produced by running quicksort on the evolving input. We present a simple analysis showing a bound of $O(n \log n)$ on the number of blocking pairs, which is better than the trivial bound of order n^2. Next, we analyze the Gale-Shapley algorithm when evolution events only occur on one side and the preference lists are uniformly random permutations; in particular the preference lists on the A side are chosen uniformly at random, and the preference lists on the B-side are subject to evolution events.[3] We present a simple analysis showing an $O(\log n)$ bound on the number of blocking pairs for this special case.

3.1 A simple algorithm

Our first algorithm ignores evolution of preference lists and runs the standard Gale-Shapley algorithm to produce a matching. More specifically, it first obtains the preferences lists for all $2n$ agents using the QUICKSORT algorithm of [1] (*i.e.*, using Algorithm 1) and then produces a matching by running the Gale-Shapley algorithm on these lists (ignoring the fact that these lists are only *approximately* correct).

[3] Note that after sufficiently many time-steps (though still polynomial) the evolution events ensure that all permutations are uniformly random. This follows from analyzing the mixing time of the corresponding Markov chain over permutations. See for example the book [7].

We show that this simple algorithm maintains a matching with at most $O(n \log n)$ blocking pairs. Note that the number of blocking pairs is trivially at most n^2. We further argue that improving the bound would require new ideas that either go around Lemma 4 (Lemma 6 of [1]) or improve the analysis in a substantial way.

Algorithm 2 runs in perpetuity. The matching M is maintained as the output until the new matching based on the newly sorted preference lists can be computed.

Algorithm 2 : Simple dynamic stable matching

```
1: while TRUE do
2:     $\{\tilde{\pi}_z : z \in A \cup B\} \leftarrow$ SEQUENTIALSORT($\{\pi_z : z \in A \cup B\}$)   ▷ Calling Algorithm 1
3:     return Gale-Shapley matching $M$ on the (approximately) sorted lists $\{\tilde{\pi}_z : z \in A \cup B\}$
```

▶ **Theorem 6.** *For a sufficiently large constant c_0 and any time-step $t \geq c_0 n^2 \log n$, Algorithm 2 maintains a matching with $O(n \log n)$ blocking pairs in expectation and with high probability.*

Proof. We consider the number of blocking pairs at time-step $T \leq t$ when the current matching M was computed. In the following discussion, we use the following notation: for $x \in A, y \in B$, if $M(x) = y$, then $M(y) = x$ (rather than M^{-1}). At time-step T, for each $z \in A \cup B$, define the indicator function $I_z(w)$ to be 1 when $M(z) \prec_{\tilde{\pi}_z} w$ and $w \prec_{\pi_z} M(z)$ and 0 otherwise. (We don't explicitly use superscripts on the preference lists π as time-step T is fixed until specified otherwise.) By Lemma 4, in expectation and with high probability,

$$\sum_w I_z(w) = \left|\{w : w \prec_{\pi_z} M(z) \text{ and } M(z) \prec_{\tilde{\pi}_z} w\}\right| = O(\log n), \tag{1}$$

If a pair (x, y) is blocking at time-step T, then $x \prec_{\pi_y} M(y)$ and $y \prec_{\pi_x} M(x)$. Assume $x \prec_{\tilde{\pi}_y} M(y)$ and $y \prec_{\tilde{\pi}_x} M(x)$. Since $y \prec_{\tilde{\pi}_x} M(x)$, x must have proposed to y at some point during the execution of the Gale-Shapley algorithm. By the properties of the Gale-Shapley algorithm, y should have been matched to an element of A with rank according to $\tilde{\pi}_y$ at least as high as the rank of x in $\tilde{\pi}_y$. This contradicts $x \prec_{\tilde{\pi}_y} M(y)$. It follows that either $M(y) \prec_{\tilde{\pi}_y} x$ or $M(x) \prec_{\tilde{\pi}_x} y$. Define $U(x, y)$ to be 1 when (x, y) is blocking and 0 otherwise. We have argued that

$$U(x, y) \leq I_x(y) + I_y(x).$$

By the union bound, Equation 1 holds for every $z \in A \cup B$ with high probability. Summing over all pairs (x, y) and applying the union bound again

$$\sum_{x,y} U(x, y) \leq \sum_{x,y} I_x(y) + I_y(x) = \sum_x \Big(\sum_y I_x(y)\Big) + \sum_y \Big(\sum_x I_y(x)\Big) = O(n \log n),$$

in expectation and with high probability.

Next, let $t' = t - T$. We need to account for blocking pairs that may have arisen during these t' time-steps. First, we observe that by Lemma 2 and union bound, with high probability $t' = O(n^2 \log n)$ (as otherwise another matching more up-to-date than the one at time-step T would be available). During the t' time-steps from T to t, evolution may create a blocking pair only if the swap decreases the rank of $M(z)$ in π_z, for some $z \in A \cup B$. Therefore, each step of the evolution introduces—independently—a new blocking pair with

A-side lists π

1	3	2	1	4	5	6	7
2	4	3	2	5	6	7	1
3	5	4	3	6	7	1	2
4	6	5	4	7	1	2	3
5	7	6	5	1	2	3	4
6	1	7	6	2	3	4	5
7	2	1	7	3	4	5	6

B-side lists π

1	6	7	1	5	4	3	2
2	7	1	2	6	5	4	3
3	1	2	3	7	6	5	4
4	2	3	4	1	7	6	5
5	3	4	5	2	1	7	6
6	4	5	6	3	2	1	7
7	5	6	7	4	3	2	1

A-side lists $\tilde{\pi}$

1	1	2	3	4	5	6	7
2	2	3	4	5	6	7	1
3	3	4	5	6	7	1	2
4	4	5	6	7	1	2	3
5	5	6	7	1	2	3	4
6	6	7	1	2	3	4	5
7	7	1	2	3	4	5	6

B-side lists $\tilde{\pi}$

1	1	7	6	5	4	3	2
2	2	1	7	6	5	4	3
3	3	2	1	7	6	5	4
4	4	3	2	1	7	6	5
5	5	4	3	2	1	7	6
6	6	5	4	3	2	1	7
7	7	6	5	4	3	2	1

Figure 1 Instances π and $\tilde{\pi}$ demonstrating tightness of Algorithm 2.

probability at most $\alpha/(n-1)$. The expected number of blocking pairs introduced is therefore at most $O(n \log n)$ for $\alpha = \Theta(1)$ (and assuming $t' = O(n^2 \log n)$). The result follows by a simple application of the Chernoff and union bounds. ◀

▶ Remark 7. We present a pair of instances to the stable matching problem with preference lists π_z and $\tilde{\pi}_z$ for $z \in A \cup B$, for which the conclusion of Lemma 4 is satisfied, *i.e.*, for any z the number of pairs (i,j) that are ordered differently in π_z and $\tilde{\pi}_z$ are $O(\log n)$. We then show that a matching that is stable with respect to $\tilde{\pi}$ as preference lists has $\Omega(n \log n)$ blocking pairs with respect to π. Thus, it follows that using the QUICKSORT algorithm of Anagnostopoulos *et al.* [1] and its analysis as a blackbox will not result in a stronger result than the one provided in Theorem 6.

First, we define the preference lists $\tilde{\pi}_z$ for $z \in A \cup B$. Let $A = B = [n]$. Then for $x \in A$, the preference list (ranking) $\tilde{\pi}_x$ is defined as $x, x+1, \ldots, n, 1, 2, \ldots, x-1$. On the other hand for $y \in B$, the preference list (ranking) $\tilde{\pi}_y$ is defined as $y, y-1, \ldots, 1, n, n-1, \ldots, y+1$. The rankings π_z, $z \in A \cup B$, are now defined as follows: let k be some parameter, π_z simply as the elements at rank 1 and k swapped. Figure 1 shows an example with $n = 7$ and $k = 3$. Clearly, when $k = \Theta(\log n)$, π_z and $\tilde{\pi}_z$ satisfy the conclusion of Lemma 4. Yet, it is easy to see that $M(i) = i$ is a stable matching for the preference lists $\tilde{\pi}_z$, $z \in A \cup B$, and for this matching with respect to the preference lists π_z, $z \in A \cup B$, every pair (i,j) with $0 < j - i < k$ is a blocking pair.

3.2 One-sided evolution

In this section we analyze how the Gale-Shapley algorithm performs when the initial preference lists are random, but there is no evolution on the lists of the elements in A. Furthermore, we are going to assume that the algorithm knows each permutation in $\{\pi_x : x \in A\}$. We call this setting *one-sided evolution.*

In this setting, the standard Gale-Shapley algorithm is implemented (basic pseudocode is shown in Algorithm 3). Note that the only time the preference lists on the B-side are used is in Line 11. Thus, it is only for these steps that we need to query the input (since the preference lists on the A-side are known to the algorithm). Thus, the number of queries made by the algorithm is bounded by the number of proposals. It was already observed that the number of proposals made in a random instance of the stable matching problem is $O(n \log n)$. The actual algorithm keeps implementing the Gale-Shapley algorithm from scratch after completion. The matching from the previous completed run is used as the current matching. We prove the following result for the one-sided evolution setting.

Algorithm 3 Gale-Shapley Algorithm

```
1: $M \leftarrow \phi$                                          ▷ Initialize empty matching
2: for $x \in A$ do
3:     new_match ← False
4:     $p \leftarrow x$
5:     while new_match = False do
6:         $y \leftarrow$ first as yet unproposed as per $\pi_p$
7:         if $M(y)$ not yet set then
8:             $M(p) \leftarrow y$
9:             $M(y) \leftarrow p$
10:            new_match = True
11:        else if $p \prec_{\pi_y} M(y)$ then                     ▷ $y$ prefers $p$ to $M(y)$
12:            $p' \leftarrow M(y)$
13:            $M(y) \leftarrow p$
14:            $M(p) \leftarrow y$
15:            $p \leftarrow p'$
16: return $M$
```

▶ **Theorem 8.** *For a sufficiently large constant c_0 and any time-step $t \geq c_0 n \log n$, the Gale-Shapley algorithm (repeatedly run and using the matching of the last completed run as the output) under one-sided evolution maintains a matching with at most $O(\log n)$ blocking pairs in expectation and with high probability.*

Proof. To prove the bound on the number of blocking pairs, call an evolution event on y's list *critical* if it involves the *then* match of y. Suppose that after the algorithm terminates, $y \in B$ is involved in k blocking pairs $(x_1, y), \ldots, (x_k, y)$. We observe that each one of the $x_1, \ldots, x_k$ was involved in at least one critical evolution step. To see this note that if (x, y) is blocking, then x proposed to y during the execution of the algorithm and got rejected subsequently (because π_x didn't change and $y \prec_{\pi_x} M(x)$). But since at the end of the execution it forms a blocking pair, it must ranked higher than $M(y)$. This is only possible if x was swapped with the *then* match of y in some evolution event during the execution of the algorithm.

Given this observation, we estimate the number of blocking pairs by estimating the number of critical evolution steps. Note that an evolution step is critical with probability at most $2\alpha/(n-1)$ (at most $2n$ out of $n(n-1)$ pairs involve the matching). Let T be a random variable equal to the number of proposals before the algorithm outputs a matching and label the corresponding time-steps as $1, 2, \ldots, T$. For each step k, let X_k be a Boolean random variable that is equal to 1 if at the time-step labeled k some evolution event was critical.

First note that from a coupon-collecting argument as in Theorem 1 it follows that $T = O(n \log n)$ in expectation and with high probability. This is because for that argument the distribution of $\{\pi_y : y \in B\}$ is irrelevant and $\{\pi_x : x \in A\}$ being random permutations suffices. Therefore, we may fix an appropriately large constant C so that $T \leq Cn \log n$ with high probability and let $m = Cn \log n$. As noted above, at any given step and given any information from the previous steps, an evolution step is critical with probability at most $2\alpha/(n-1)$; thus, by Theorem 5,

$$\mathbb{P}\Big[\sum_{k=1}^{m} X_k > 2C \log n\Big] = O(n^{-C}),$$

for sufficiently large C. We have

$$\begin{aligned}\mathbb{P}\Big[\sum_{k=1}^{T} X_k > 2C\log n\Big] &\leq \mathbb{P}\big[T > Cn\log n\big] + \mathbb{P}\Big[\Big(\sum_{k=1}^{T} X_k > 2C\log n\Big) \wedge \big(T \leq Cn\log n\big)\Big]\\ &\leq \mathbb{P}\big[T > Cn\log n\big] + \mathbb{P}\Big[\sum_{k=1}^{m} X_k > 2C\log n\Big].\end{aligned}$$

By the observation at the beginning of this paragraph the claimed bound holds with high probability. The bound on the expectation follows by noting that there can be at most n^2 blocking pairs.

Finally, note that there will be at most $O(n \log n)$ time-steps before a new matching is computed. As in the final paragraph in the proof of Theorem 6, one can show that evolution cannot produce more than $O(\log n)$ blocking pairs in these many steps. ◀

4 General Case: Improved algorithm

We now consider the general setting where the preference lists on both sides may be evolving. We present a modified version of the Gale-Shapley algorithm that takes advantage of Lemma 4 (Lemma 6 of [1]) and maintains a stable matching with at most $O((\log n)^2)$ blocking pairs. The algorithm consists of two separate processes that run in an interleaved fashion: the *sorting process* on even time-steps and the *matching process* on odd ones. The sorting process is basically a call to SequentialSort($\{\pi_x \mid x \in A\}$) that produces *approximately sorted* preference lists on the A side, $\{\tilde{\pi}_x \mid x \in A\}$. The algorithm runs in perpetuity, in the sense that as soon as it terminates it restarts, though the copies $\{\tilde{\pi}_x \mid x \in A\}$ from the previous execution are retained to be used by the stable matching process. Initially, the $\tilde{\pi}_x$ are set to be random permutations, thus, for the first $O(n^2 \log n)$ steps, until one run of the sorting process is complete, the matching output by the algorithm will be garbage.

Algorithm 4 : Interleaving Sorting and Matching

```
1: for $t = 1, 2, \dots$ do
2:     if $t$ is Even then
3:         Perform query for Algorithm 1
4:     else if $t$ is Odd then
5:         Perform query for Algorithm 5
```

We note that what is counted here is only time-steps; making queries is the bottleneck. Additional computations required by the algorithm can be performed in between time-steps at no additional cost.

The sorting process performs queries only during even steps and its purpose it to keep the preference list of each $x \in A$ approximately sorted, where by approximately sorted we meant that the conclusion of Lemma 4 holds.

The matching process performs queries during odd steps. Our stable matching algorithm, which is a variant of the Gale-Shapley algorithm, is presented in Algorithm 5. Note that the $\{\tilde{\pi}_x \mid x \in A\}$ used in Algorithm 5 are *static* and are the output of the latest completed run of Algorithm 1. However, the comparisons made are all on dynamic data. The difference from the standard Gale-Shapley algorithm is that whenever some $x \in A$ is about to make a proposal, first the *best* $y \in B$ among the $O(\log n)$ highest ranked as per the ranking $\tilde{\pi}_x$ that have not yet rejected x is found. Note however, that the best is with respect to the dynamic (current) preference list π_x (otherwise, it would be trivial since $\tilde{\pi}_x$ is static). This operation is basically the algorithm to find the minimum element, which can be implemented in $O(\log n)$ time using only comparison queries (see Section 3 in [1]). We don't need to use any particular result regarding finding the minimum element; instead, we incorporate the errors that may have occurred while finding the minimum due to the dynamic nature of the data, into our stable matching analysis directly.

Algorithm 5 : Modified Gale-Shapley Algorithm

```
1: $M \leftarrow \emptyset$
2: for $x \in A$ do
3:     new_match ← FALSE
4:     $p \leftarrow x$
5:     while new_match = FALSE do
6:         $S \leftarrow C \log n$ highest-ranked, not-yet-proposed elements in $B$ per $\tilde{\pi}_x$
7:         $y \leftarrow$ best($S$)                    ▷ Best with respect to dynamic $\pi_x$
8:         if $M(y)$ not yet set then
9:             $M(p) \leftarrow y$
10:            $M(y) \leftarrow p$
11:            new_match ← TRUE
12:        else if $p \prec_{\pi_y} M(y)$ then
13:            $p' \leftarrow M(y)$
14:            $M(y) \leftarrow p$
15:            $M(p) \leftarrow y$
16:            $p \leftarrow p'$
17: return $M$
```

We first describe the high-level idea of the proof. The sorting process needs $O(n^2 \log n)$ comparisons with high probability. The approximations $\{\tilde{\pi}_x : x \in A\}$ of $\{\pi_x : x \in A\}$ that are being computed are used by the modified Gale-Shapley algorithm. By Lemma 4 we are able to claim that for any element u in the preference list of any $x \in A$, the number of pairs (u, v) that are ordered differently in $\tilde{\pi}_x$ and π_x are $O(\log n)$. Therefore, when x is about to propose it suffices to look among $O(\log n)$ elements in $\tilde{\pi}_x$ to find the y to which the proposal will be made. Since the matching process is expected to make $O(n \log n)$ proposals, it is expected to require $O(n(\log n)^2)$ comparisons. It turns out that, during these steps, evolution creates a blocking pair with probability at most α/n. Therefore we expect $O((\log n)^2)$ blocking pairs.

We now provide the details of the proof. In order to bound the number of blocking pairs, it is crucial that during the matching process not too many queries are made, or alternatively that not too many proposals are made. We therefore need an analog of Theorem 1. To apply the coupon-collecting argument from the proof of that theorem we prove the following lemma.

▶ **Lemma 9.** *Provided π_x was chosen uniformly at random at time-step 0 and only comparison queries are made, the element y chosen at line 7 of Algorithm 5 is a random element from the subset of B to which x has not by that point made any proposals.*

Proof. The proof of the lemma relies on the fact that the dynamic quicksort algorithm used to obtain $\tilde{\pi}_x$ for $x \in A$ and the procedure used to find the best element in line 7 of Algorithm 5 only use comparison queries.

Let π_x be the preference list of x before the first comparison is queried. Fix an arbitrary sequence of evolution steps that occurs during the computation of y. Suppose that given these choices of nature, $y = \pi_x(k)$. Then, given the same evolution steps, for any other permutation π'_x, $y = \pi'_x(k)$. Since π_x is a random permutation, $y = \pi_x(k)$ is a random element of σ_x. ◀

▶ Remark 10. We remark that the requirement on the implementation of QUICKSORT and `best` using only comparison queries is necessary and the lemma does not hold for an arbitrary algorithm.

▶ **Lemma 11.** *The number of proposals during one execution of the matching process is $O(n \log n)$ in expectation and with high probability.*

Proof. Suppose x proposes to y and at that point k elements of B are unmatched. Note that it must be the case that x has not proposed to any of these elements (otherwise, they would not be currently unmatched). Thus, by the previous lemma, each of these k elements receives a proposal with probability at least $1/n$. The stated bound follows from the analysis of coupon collector's problem as in Theorem 1. ◀

As in the one-sided setting, the analysis will rely on estimating the occurrence of a specific kind of critical evolution steps. In the present case the definition of a critical evolution step is a little more involved than its one-sided counterpart.

▶ **Definition 12.** An evolution event on the preference list π_z of $z \in A \cup B$ is *critical* if one of the following holds:

1. An evolution event involves a swap of the *then* match of z, $M(z)$.
2. If $z \in A$, the evolution event involves swapping the *then* best element as per π_z to which z has not yet proposed.

The following lemma establishes the link between the critical evolution steps and the number of blocking pairs.

▶ **Lemma 13.** *Assume that for the duration of one run of the matching process, the preference lists $\{\pi_z \mid z \in A \cup B\}$ satisfy the conditions of Lemma 4, and suppose that (x, y) is a blocking pair with respect to the returned matching. Then there was a critical evolution event on the preference list of at least one of x and y during the execution of the matching process.*

Proof. First consider the case that x proposed to y during the execution of the matching process. It follows that y rejected x at some point in favor of some other element. When x was rejected, the *then* $M(y)$ satisfies $M(y) \prec_{\pi_y} x$. Subsequently, $M(y)$ may change but has to become better, unless there was a swap that involves the *then* $M(y)$, which is a critical event on π_y. Since, we know that in the end $x \prec_{\pi_y} x'$, where x' is the final match of y, there must have been some evolution event where x was swapped with the *then* match of y. Thus, by part 1 of Definition 12, a critical evolution event occurred on π_y.

On the other hand, suppose x never proposed to y, and let y' be the final match of x. Suppose that when x proposed to y', $y \prec_{\pi_x} y'$; it follows that `best` on line 7 of Algorithm 5 failed to return the best element to which x had not yet proposed. Since, we are assuming that $\tilde{\pi}_x$ is a sufficient approximation of π_x, it must be because the actual *best* element was swapped at least once while `best` was being executed. Thus, by part 2 of Definition 12, a critical evolution event occurred on π_x. Finally, if when x proposed to y', it was the case that $y' \prec_{\pi_x} y$, but (x, y) is a blocking pair, it must be that y' was involved in a swap subsequently leading to a critical event involving x's the *then* match. ◀

▶ **Theorem 14.** *Provided the initial preference lists are drawn randomly,*[4] *for all $z \in A \cup B$, for a sufficiently large constant c_0 and any time-step $t \geq c_0 n^2 \log n$, Algorithm 5 maintains a matching with at most $O((\log n)^2)$ blocking pairs in expectation and with high probability.*

Proof. As a result of Lemma 13, we can estimate the number of blocking pairs by estimating the number of critical evolution steps. Let T be a random variable equal to the number of queries of Algorithm 5 before it outputs a matching and label the corresponding time-steps as $1, 2, \dots, T$. For each step k, let X_k be a Boolean random variable that is equal to 1 if at the time-step labeled k the evolution was critical. Furthermore, denote by $\mathcal{E}$ the event that during these T time-steps the lists were approximately sorted. By Lemma 4, the event $\mathcal{E}$ occurs with high probability.

By Lemma 11 it follows that $T = O(n(\log n)^2)$ in expectation and with high probability, since we are wasting $O(\log n)$ queries per proposal. Therefore, we may fix an appropriately large constant C so that $T \leq Cn(\log n)^2$ with high probability and let $m = Cn(\log n)^2$. Note that—given any history of evolution steps—an evolution step is critical with probability at most $O(\alpha/n)$, since for each $z \in A \cup B$ there is a constant number of elements that evolution has to swap in order to be critical. Thus, by Theorem 5,

$$\mathbb{P}\Big[\sum_{k=1}^{m} X_k > 2C(\log n)^2\Big] = O(n^{-C}),$$

for sufficiently large C. We have

$$\begin{aligned}\mathbb{P}\Big[\sum_{k=1}^{T} X_k > 2C(\log n)^2\Big] &\leq \mathbb{P}\big[T > Cn(\log n)^2\big] + \mathbb{P}[\bar{\mathcal{E}}] \\ &\quad + \mathbb{P}\Big[\Big(\sum_{k=1}^{T} X_k > 2C(\log n)^2\Big) \wedge \big(T \leq Cn(\log n)^2\big) \wedge \mathcal{E}\Big] \\ &\leq \mathbb{P}\big[T > Cn(\log n)^2\big] + \mathbb{P}[\bar{\mathcal{E}}] + \mathbb{P}\Big[\sum_{k=1}^{m} X_k > 2C(\log n)^2\Big].\end{aligned}$$

It follows that the claimed bound holds with high probability. The bound on the expectation follows by noting that there can be at most n^2 blocking pairs. ◀

Acknowledgments. For earlier discussions, F.M. would like to thank Marcos Kiwi who, among other things, introduced him to the line of works on evolving data and shared preliminary thoughts on the possibility of computing stable matchings in this context. We also thank a reviewer for (minor) corrections in our main theorem.

References

1 A. Anagnostopoulos, R. Kumar, M. Mahdian, and E. Upfal. Sorting and selection on dynamic data. *Theoretical Computer Science*, 412(24):2564–2576, 2011. Selected Papers from 36th International Colloquium on Automata, Languages and Programming. doi:10.1016/j.tcs.2010.10.003.

[4] This is not actually required, since after sufficiently long (though still polynomial) time, all the preference lists will be close to random due to a mixing time argument on the set of permutations.

2 A. Anagnostopoulos, R. Kumar, M. Mahdian, E. Upfal, and F. Vandin. Algorithms on evolving graphs. In *Proc. of 3rd Innovations in Theoretical Computer Science*, 2012.

3 D. Eppstein, Z. Galil, and G. F. Italiano. Dynamic graph algorithms. In M. Atallah, editor, *Algorithms and Theory of Computation Handbook*, chapter 8. CRC Press, 1999.

4 D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

5 M. Gupta and R. Peng. Fully dynamic (1+ e)-approximate matchings. In *Proc. of 54th IEEE Foundations of Computer Science*, pages 548–557, Oct 2013. `doi:10.1109/FOCS.2013.65`.

6 D. Knuth. *Stable Marriage and Its Relation to Other Combinatorial Problems: An Introduction to the Mathematical Analysis of Algorithms.* CRM proceedings & lecture notes. American Mathematical Society, 1997.

7 David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov Chains and Mixing Times.* American Mathematical Society, 2009.

8 C. McDiarmid. Concentration. In M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, editors, *Probabilistic Methods for Algorithmic Discrete Mathematics*, volume 16 of *Algorithms and Combinatorics*, pages 195–248. Springer Berlin Heidelberg, 1998. `doi:10.1007/978-3-662-12788-9_6`.

9 R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge International Series on Parallel Computation. Cambridge University Press, 1995.

10 O. Neiman and S. Solomon. Simple deterministic algorithms for fully dynamic maximal matching. In *Proc. of 45th ACM Symposium on Theory of Computing*, pages 745–754, 2013. `doi:10.1145/2488608.2488703`.

11 K. Onak and R. Rubinfeld. Maintaining a large matching and a small vertex cover. In *Proc. of 42nd ACM Symposium on Theory of Computing*, pages 457–464, 2010. `doi:10.1145/1806689.1806753`.

12 L. B. Wilson. An analysis of the stable marriage assignment algorithm. *BIT Numerical Mathematics*, 12(4):569–575, 1972.

An $\widetilde{O}(n)$ Queries Adaptive Tester for Unateness[*]

Subhash Khot[1] **and Igor Shinkar**[2]

1 **Courant Institute of Mathematical Sciences, New York University, USA**
`khot@cims.nyu.edu`

2 **Courant Institute of Mathematical Sciences, New York University, USA**
`ishinkar@cims.nyu.edu`

Abstract

We present an adaptive tester for the unateness property of Boolean functions. Given a function $f : \{0,1\}^n \to \{0,1\}$ the tester makes $O(n\log(n)/\varepsilon)$ adaptive queries to the function. The tester always accepts a unate function, and rejects with probability at least 0.9 if a function is ε-far from being unate.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases property testing, boolean functions, unateness

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.37

1 Introduction

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be *unate* if for every $i \in [n]$ it is either the case that f is monotone non-increasing in the i'th coordinate, or f is monotone non-decreasing in the i'th coordinate. In this work we present an adaptive tester for the unateness property that makes $O(n\log(n)/\varepsilon)$ adaptive queries to a given function. The tester always accepts a unate function, and rejects with probability at least 0.9 any function that is ε-far from being unate.

Testing unateness has been studied first in the paper of Goldreich et al. [10], where the authors present a non-adaptive tester for unateness that makes $O(n^{1.5}/\varepsilon)$ queries. The tester in [10] is the so-called "edge tester", that works by querying the function on the endpoints of $O(n^{1.5}/\varepsilon)$ uniformly random edges of the hypercube, i.e., uniformly random pairs (x, y) that differ in one coordinate, and checking that there are no violations to the unateness property.

The notion of unateness generalizes the notion of monotonicity. Recall that a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be monotone if $f(x) \leqslant f(y)$ for all $x \prec y$, where $\prec$ denotes the natural partial order on Boolean strings, namely, $x \prec y$ if $x_i \leqslant y_i$ for all $i \in [n]$. Since the original paper of [10] there has been a lot of research concerning the problem of testing monotonicity of Boolean functions, as well as many closely related problems, such as testing monotonicity on functions with different (non-Boolean) domains [8, 9, 4, 13, 5, 7, 6, 1, 2], culminating in a recent result of [11], which gives a $\widetilde{O}(\sqrt{n}/\varepsilon^2)$-query non-adaptive tester for monotonicity. In this paper we will use the monotonicity tester of [10], which has a better dependence on ε.

▶ **Theorem 1** (Testing Monotonicity [10]). *For any proximity parameter $\varepsilon > 0$ there exists a non-adaptive tester for the monotonicity property that given a function $f : \{0,1\}^n \to \{0,1\}$ the tester makes $O(n/\varepsilon)$ queries to the function. The tester always accepts a monotone*

[*] Research supported by NSF grants CCF 1422159, 1061938, 0832795 and Simons Collaboration on Algorithms and Geometry grant.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016).
Editors: Klaus Jansen, Claire Matthieu, José D. P. Rolim, and Chris Umans; Article No. 37; pp. 37:1–37:7
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

function, and if a function is ε-far from being monotone, the tester finds a violation to monotonicity with probability at least 0.99*.*

We remark that the monotonicity testers analyzed in [10, 5, 7, 11] are all pair testers that pick pairs $x \prec y$ according to some distribution, and check that the given function f does not violate monotonicity on this pair, i.e., checks that $f(x) \leqslant f(y)$. It is not clear whether a variant of such tester can be applied for testing unateness, since the function can be monotone increasing in some of the coordinates where x and y differ, and monotone decreasing in others.

1.1 Our result

In this paper we prove the following theorem.

▶ **Theorem 2.** *For any proximity parameter $\varepsilon > 0$ there exists an adaptive tester for the unateness property, that given a Boolean function $f : \{0,1\}^n \to \{0,1\}$ makes $O(n\log(n)/\varepsilon)$ adaptive queries to f. The tester always accepts a unate function, and rejects with probability at least 0.9 any function that is ε-far from being unate.*

The tester works as follows. Given a function $f : \{0,1\}^n \to \{0,1\}$, the tester first finds a subset of coordinates $T \subseteq [n]$ such that the function is essentially independent of the coordinates outside T. Specifically, it finds a subset of coordinates $T \subseteq [n]$ such that $\mathbb{E}_{z\in\{0,1\}^T}[\mathrm{Var}_{w\in\{0,1\}^{[n]\setminus T}}[f(z_T \circ w_{\overline{T}})]]$ is small, i.e., if we pick $x, y \in \{0,1\}^n$ that are equal on their coordinates in T uniformly at random, then with high probability we will have $f(x) = f(y)$. Furthermore, for each $i \in T$ the tester will find an edge $(x, x + e_i)$ in the hypercube such that $f(x) \neq f(x+e_i)$ (where e_i is the unit vector with 1 in the i'th coordinate) Querying f on these two points gives a "direction" for monotonicity for each coordinate in T.

In the second part of the tester, we define a function that depends only on the coordinates in T by fixing the variables outside T uniformly at random. We then apply the monotonicity tester from Theorem 1 on this function with respect to the directions obtained for the coordinates in T in the previous step, and output the answer of this tester. For the analysis, we use the fact that *on average* the restricted function is close to the original function f, and hence is far from being unate. In particular, it is far from being a monotone function with respect to the directions for the coordinates in T obtained in the first step. Hence a monotonicity tester with high probability will find a violation of monotonicity in these directions, which will serve as evidence that the function is not unate.

2 Preliminaries

▶ **Definition 3.** For two Boolean functions $f, g : \{0,1\}^n \to \{0,1\}$ defined the distance between them as $\mathsf{distance}(f,g) = \Pr_{x\in\{0,1\}^n}[f(x) \neq g(x)] = 2^{-n}|\{x \in \{0,1\}^n : f(x) \neq g(x)\}|$. We say that f is ε-far from a collection of functions $\mathcal{P}$ if for any $g \in \mathcal{P}$ it holds that $\mathsf{distance}(f,g) \geqslant \varepsilon$.

▶ **Definition 4.** A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be *monotone non-decreasing* or simply *monotone* if $f(x) \leqslant f(y)$ for all $x \prec y$, where $\prec$ denotes the natural partial order on Boolean strings i.e., $x \prec y$ if $x_i \leqslant y_i$ for all $i \in [n]$. In other words, f is monotone if for every $i \in [n]$ the function f is monotone non-decreasing in the i'th coordinate.

For directions $B = (b_i \in \{up, down\} : i \in [n])$ let the partial order $\prec_B$ be defined as $x \prec_B y$ if for all $i \in [n]$ such that $b_i = up$ it holds that $x_i \leqslant y_i$ and for all for all $i \in [n]$ such that $b_i = down$ it holds that $x_i \geqslant y_i$. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be *monotone with respect to the directions* $B = \{b_i \in \{up, down\} : i \in [n]\}$ if $f(x) \leqslant f(y)$ for all $x \prec_B y$.

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be *unate* if it is monotone with respect to some directions, i.e., if for every $i \in [n]$ it is either the case that f is monotone non-increasing in the i'th coordinate, or f is monotone non-decreasing in the i'th coordinate.

Next we make definitions related to restrictions of Boolean functions by fixing some of the coordinates.

▶ **Definition 5.** Given a string $x \in \{0,1\}^n$ and a subset of coordinates $T \subseteq [n]$ denote by x_T the substring of x whose coordinates are indexed by T. Given two strings $x, y \in \{0,1\}^n$ and two disjoint subsets of coordinates $S, T \subseteq [n]$ denote by $x_T \circ y_S$ the string z whose coordinates are indexed by $T \cup S$ with $z_i = x_i$ if $i \in T$ and $z_i = y_i$ if $i \in S$.

▶ **Definition 6.** Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. For a subset of coordinates $T \subseteq [n]$ and $w \in \{0,1\}^{[n]\setminus T}$ denote by $f_{T,w} : \{0,1\}^n \to \{0,1\}$ the Boolean function defined as $f_{T,w}(z) = f(z_T \circ w_{[n]\setminus T})$. That is, for each $w \in \{0,1\}^{[n]\setminus T}$ the function $f_{T,w}$ depends only on the coordinates in T.

▶ **Definition 7.** Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $T \subseteq [n]$ be a subset of coordinates. Define $\mathrm{Var}_{[n]\setminus T}(f) = \mathbb{E}_{z \in \{0,1\}^T}[\mathrm{Var}_{w \in \{0,1\}^{[n]\setminus T}}[f(z_T \circ w_{\overline{T}})]]$.

This quantity has been used before, e.g., in [12, 3]. It measures how much f is depends on the coordinates outside T. In particular, if f depends only on the coordinates in T, (i.e., is independent of the coordinates in $[n] \setminus T$) then $\mathrm{Var}_{[n]\setminus T}(f) = 0$.

The following proposition is straightforward from the definition.

▶ **Proposition 8.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. and let $T \subseteq [n]$ be a subset of coordinates. Pick $x, y \in_R \{0,1\}^n$ such that $x_i = y_i$ for all $i \in T$ and $\{x_i, y_i \in \{0,1\} : i \in [n] \setminus T\}$ are chosen independently and uniformly at random. Then $Var_{[n]\setminus T}(f) = \Pr[f(x) \neq f(y)]$.*

3 Proof of Theorem 2

Below we present our tester for the unateness property. The tester uses a subroutine called Find an influential coordinate which works as follows. It gets an oracle access to a Boolean function $f : \{0,1\}^n \to \{0,1\}$, and a subset of the coordinates $T \subseteq [n]$, which is given explicitly. The subroutine outputs either $\perp$ or some $i^* \in [n] \setminus T$ and $b \in \{up, down\}$ such that there exist $x, y \in \{0,1\}^n$ that differ only in the i^*'th coordinate, satisfy $f(x) \neq f(y)$, and b is the orientation of f along the edge (x, y).

The subroutine Find an influential coordinate has the guarantee that if f has some non-negligible dependence on the coordinates outside T, then with some non-negligible probability the subroutine will return some $i^* \in [n] \setminus T$ and $b \in \{up, down\}$ as above. This is done by picking independently and uniformly at random two inputs $x, y \in \{0,1\}^n$ that are equal on their coordinates in T such that $f(x) \neq f(y)$, and then using "binary search" in order to decrease $\mathsf{distance}(x, y)$ to 1, while preserving the invariant that $f(x) \neq f(y)$. Specifically, given $x, y \in \{0,1\}^n$ such that $f(x) \neq f(y)$ we pick an arbitrary $z \in \{0,1\}^n$ such that if $V = \{i \in [n] : x_i \neq y_i\}$ is the set of the coordinates where $x_i = y_i$, then $z_i = x_i$ for all $i \in [n] \setminus V$, and $\mathsf{distance}(z, x) = \lfloor |V|/2 \rfloor$ and $\mathsf{distance}(z, y) = \lceil |V|/2 \rceil$. Since $f(x) \neq f(y)$, it must be the case that $f(z)$ differs from either $f(x)$ or $f(y)$. We then update either x or y to be z so that $f(x) \neq f(y)$. This clearly decreases $\mathsf{distance}(x, y)$ by roughly a multiplicative

```
procedure FIND AN INFLUENTIAL COORDINATE($f : \{0,1\}^n \to \{0,1\}, T$)
    Pick $x, y \in_R \{0,1\}^n$ independently and uniformly at random such that $x_T = y_T$
    if $f(x) = f(y)$ then
        return $\perp$
    else ($f(x) \neq f(y)$)
        repeat
            $U \leftarrow \{i \in [n] : x_i = y_i\}$
            $V \leftarrow \{j \in [n] : x_j \neq y_j\}$
            Pick an arbitrary $z_V \in \{0,1\}^V$ such that $|\{i \in V : z_i = y_i\}| = \lfloor |V|/2 \rfloor$.
            Let $z = x_U \circ z_V \in \{0,1\}^n$
            if $f(x) \neq f(z)$ then
                $y \leftarrow z$
            else ($f(y) \neq f(z)$)
                $x \leftarrow z$
            end if
        until $|V| = 1$
        Let $i^* \in [n]$ be the unique element in $V$
        Let $b \in \{up, down\}$ be the orientation of $f$ in the edge $(x, y)$
        return $(i^*, b)$
    end if
end procedure
```

```
procedure UNATENESS TESTER($f : \{0,1\}^n \to \{0,1\}$)
    Let $m = O(\frac{n}{\varepsilon})$
    Let $T = \emptyset$
    for $i = 1...m$ do
        Find an influential coordinate($f, T$)
        if returned a coordinate and a direction $(i^*, b_{i^*})$ then
            Add $i^*$ to $T$, and let $b_{i^*}$ be the corresponding direction.
        end if
    end for
    Pick $w \in \{0,1\}^{[n]\setminus T}$
    Apply the monotonicity tester on $f_{T,w}$ with respect to the directions $\{b_i : i \in T\}$
    Return the output of the monotonicity tester.
end procedure
```

factor of 2, and so, by repeating this at most $\log(n)$ times we obtain x and y that satisfy $f(x) \neq f(y)$ and differ in exactly one coordinate.

For the proof of Theorem 2 we need the following two claims.

▶ **Claim 9.** *Let $c > 0$ be a small constant and let $m = \frac{2n}{c\varepsilon}$ be the number of iterations of the* for *loop in the Unateness tester. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $T \subseteq [n]$ be the set in the Unateness tester after m iterations of the* for *loop. Then, with high probability the set T satisfies*

$$Var_{[n]\setminus T}(f) < c\varepsilon.$$

Proof. Note that if in some iteration we have a subset of coordinates $T \subseteq [n]$ such that $\mathrm{Var}_{[n]\setminus T}(f) \geqslant c\varepsilon$, then, by Proposition 8 the variables x and y chosen in line 2 of Find an influential coordinate(f, T) will satisfy $f(x) \neq f(y)$ with probability at least $c\varepsilon$. Having such x and y, let $U \subseteq [n]$ be the coordinates where x and y are equal, and let $V \subseteq [n]$ be the coordinates where the two strings differ. Then, in each iteration the procedure chooses z at random, such that it agrees with x and y in the coordinates where they equal, and updates x or y according to the value of $f(z)$, while preserving the property that $f(x) \neq f(y)$. Clearly, if $z \neq y$ and $z \neq x$, then in each step we reduce the distance between x and y, until $|V| = 1$, i.e., $y = x + e_i$ for the unique coordinate $i \in V$, which is returned by the procedure together with the orientation of the edge (x, y).

Therefore, if $m = \frac{2n}{c\varepsilon}$, then by Azuma's inequality with probability $1 - e^{-\Omega(n)}$ among the m iterations at least $\frac{c\varepsilon m}{2} = n$ iterations will have the property that either Find an influential coordinate finds a new coordinate to add to T or that $\mathrm{Var}_{[n]\setminus T}(f) \leqslant c\varepsilon$.[1] On the other hand, the function f depends on at most n coordinates, and hence, after $m = \frac{2n}{c\varepsilon}$ iterations the set T will satisfy the property

$$\mathrm{Var}_{[n]\setminus T}(f) \leqslant c\varepsilon,$$

with probability at least $1 - e^{-\Omega(n)}$, as required. ◀

▶ **Claim 10.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $T \subseteq [n]$ be such that*

$$Var_{[n]\setminus T}(f) \leqslant c\varepsilon$$

for some $\varepsilon > 0$ and $c \in (0, 1/8)$. Then, for a random $w \in \{0,1\}^{[n]\setminus T}$ it holds that

$$\Pr_{w\in\{0,1\}^{[n]\setminus T}}[\mathsf{distance}(f_{T,w}, f) \geqslant \varepsilon/2] \leqslant 8c.$$

Proof. Define the function $Maj_T : \{0,1\}^n \to \{0,1\}$ as

$$Maj_T(z) = \begin{cases} 1 & \text{if } \Pr_{w\in\{0,1\}^{[n]\setminus T}}[f(z_T \circ w_{\overline{T}}) = 1] > 0.5 \\ 0 & \text{otherwise.} \end{cases}$$

That is, Maj_T depends only on the coordinates in T. By the assumption of the claim we have that for a uniformly random $w \in \{0,1\}^{[n]\setminus T}$ it holds that

$$\begin{aligned} \mathbb{E}_{w\in\{0,1\}^{[n]\setminus T}}[\mathsf{distance}(f_{T,w}, Maj_T)] &= \mathbb{E}_{z\in\{0,1\}^T}[\Pr_{w\in\{0,1\}^{[n]\setminus T}}[f(z_T \circ w_{\overline{T}}) \neq Maj(z_T)] \\ &\leqslant \mathbb{E}_{z\in\{0,1\}^T}[2\mathrm{Var}_{w\in\{0,1\}^{[n]\setminus T}}[f(z_T \circ w_{\overline{T}})]] \\ &\leqslant 2c\varepsilon. \end{aligned}$$

[1] Formally, let $(X_i : i \in [m])$ be Bernouli random variables with $X_i = 1$ if either $\mathrm{Var}_{[n]\setminus T}(f) \leqslant c\varepsilon$ or a new coordinate is added to T in the i'th iteration, and observe that $\Pr[X_i = 1] \geqslant c\varepsilon$ for all $i \in [m]$.

Hence, by Markov's inequality

$$\Pr_w[\mathsf{distance}(f_{T,w}, Maj_T) \geqslant \varepsilon/4] \leqslant 8c.$$

On the other hand,

$$\mathsf{distance}(f, Maj_T) = \mathbb{E}_{w\in\{0,1\}^{[n]\setminus T}}[\mathsf{distance}(f_{T,w}, Maj_T)] \leqslant 2c\varepsilon \leqslant \varepsilon/4.$$

Therefore, by triangle inequality we have

$$\Pr_w[\mathsf{distance}(f_{T,w}, f) \geqslant \varepsilon/2] \leqslant \Pr_w[\mathsf{distance}(f_{T,w}, Maj_T) \geqslant \varepsilon/4] \leqslant 8c,$$

and the claim follows. ◀

Theorem 2 now follows easily from the above claims.

Proof of Theorem 2. For a small constant $c > 0$ let $m = O(\frac{n}{c\varepsilon})$ be the number of iterations of the *for* loop in the Unateness tester. Let $T \subseteq [n]$ be the set in the Unateness tester after m iterations of the for loop. By Claim 9 with probability 0.99 the set T satisfies

$$\mathrm{Var}_{[n]\setminus T}(f) \leqslant c\varepsilon.$$

Assuming that T satisfies the above, by Claim 10 if f is ε-far from being unate, then for a uniformly random $w \in \{0,1\}^{[n]\setminus T}$ it holds that $f_{T,w}$ is $\varepsilon/2$-far from being unate with probability $(1-8c)$, and in particular, it is $\varepsilon/2$ from from being monotone with respect to the directions $\{b_i : i \in T\}$. By applying the monotonicity tester on $f_{T,w}$ with w such that $f_{T,w}$ is $\varepsilon/2$-far from being unate it follows that with probability at least 0.99 the invocation of the monotonicity tester will find a violation to monotonicity of $f_{T,w}$ with respect to the directions $\{b_i : i \in T\}$. Therefore, for a sufficiently small constant $c > 0$, if f is ε-far from unate, then with probability 0.9 the tester will reject.

Finally, we analyze the query complexity of the tester. It is clear that the procedure Find an influential coordinate makes at most $O(\log(n))$ iterations, as in each iteration z differs from both x and y in at most $\lceil \mathsf{distance}(x,y)/2\rceil$ coordinates. Therefore, the total number of queries made by the tester in the for loop is $m \cdot O(\log(n))$. In addition the tester makes at most $O(n/\varepsilon)$ queries in step 11. Therefore, tester makes at most $O(n\log(n)/\varepsilon)$ queries. ◀

Acknowledgements. We are thankful to the anonymous referees for their helpful comments.

References

1 Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2016, pages 1021–1032, New York, NY, USA, 2016. ACM. doi:10.1145/2897518.2897567.

2 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners of the hypercube and the hypergrid. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:46, 2009. URL: http://eccc.hpi-web.de/report/2009/046.

3 Eric Blais. Testing juntas nearly optimally. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC'09, pages 151–158, New York, NY, USA, 2009. ACM. doi:10.1145/1536414.1536437.

4 Jop Briët, Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.

5 Deeparnab Chakrabarty and C. Seshadhri. A o(n) monotonicity tester for boolean functions over the hypercube. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 411–418, 2013. `doi:10.1145/2488608.2488660`.

6 Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 519–528, 2015. `doi:10.1145/2746539.2746570`.

7 Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 286–295, 2014. `doi:10.1109/FOCS.2014.38`.

8 Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *Randomization, Approximation, and Combinatorial Algorithms and Techniques, Third International Workshop on Randomization and Approximation Techniques in Computer Science, and Second International Workshop on Approximation Algorithms for Combinatorial Optimization Problems RANDOM-APPROX'99, Proceedings. Berkeley, CA, USA, August 8-11, 1999*, pages 97–108, 1999.

9 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC'02, pages 474–483, New York, NY, USA, 2002. ACM.

10 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000. `doi:10.1007/s004930070011`.

11 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric type theorems. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, 2015.

12 Guy Kindler and Shmuel Safra. Noise-resistant boolean-functions are juntas, 2003. Manuscript.

13 Eric Lehman and Dana Ron. On disjoint chains of subsets. *J. Comb. Theory, Ser. A*, 94(2):399–404, 2001. `doi:10.1006/jcta.2000.3148`.

A Local Algorithm for Constructing Spanners in Minor-Free Graphs*

Reut Levi[1], Dana Ron[2], and Ronitt Rubinfeld[3]

1 **MPI for informatics, Saarbrücken, Germany**
`rlevi@mpi-inf.mpg.de`
2 **School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel**
`danar@eng.tau.ac.il`
3 **CSAIL, MIT, Cambridge, MA, USA, and**
Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel
`ronitt@csail.mit.edu`

Abstract

Constructing a spanning tree of a graph is one of the most basic tasks in graph theory. We consider this problem in the setting of local algorithms: one wants to quickly determine whether a given edge e is in a specific spanning tree, without computing the whole spanning tree, but rather by inspecting the local neighborhood of e. The challenge is to maintain consistency. That is, to answer queries about different edges according to the *same* spanning tree. Since it is known that this problem cannot be solved without essentially viewing all the graph, we consider the relaxed version of finding a spanning subgraph with $(1+\epsilon)n$ edges instead of $n-1$ edges (where n is the number of vertices and ϵ is a given approximation/sparsity parameter).

It is known that this relaxed problem requires inspecting $\Omega(\sqrt{n})$ edges in general graphs (for any constant ϵ), which motivates the study of natural restricted families of graphs. One such family is the family of graphs with an excluded minor (which in particular includes planar graphs). For this family there is an algorithm that achieves constant success probability, and inspects $(d/\epsilon)^{\text{poly}(h)\log(1/\epsilon)}$ edges (for each edge it is queried on), where d is the maximum degree in the graph and h is the size of the excluded minor. The distances between pairs of vertices in the spanning subgraph G' are at most a factor of $\text{poly}(d, 1/\epsilon, h)$ larger than in G.

In this work, we show that for an input graph that is H-minor free for any H of size h, this task can be performed by inspecting only $\text{poly}(d, 1/\epsilon, h)$ edges in G. The distances between pairs of vertices in the spanning subgraph G' are at most a factor of $\tilde{O}(h\log(d)/\epsilon)$ larger than in G. Furthermore, the error probability of the new algorithm is significantly improved to $\Theta(1/n)$. This algorithm can also be easily adapted to yield an efficient algorithm for the distributed (message passing) setting.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases spanners, sparse subgraphs, local algorithms, excluded-minor

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.38

1 Introduction

Given graph $G = (V, E)$, a basic task is to find a sparse spanning subgraph G', where G' may be required to be a tree, or may be required to approximately preserve distances between

* This work was partially supported by the Israel Science Foundation grant No. 671/13 and grant No. 1536/14 and the NSF grant CCF-1420692.

vertices (i.e., have small *stretch* [14, 13]). Suppose we are interested in determining whether a given edge e belongs to the spanning subgraph $G' = (V, E')$. We can of course run an algorithm that constructs G' and check whether $e \in E'$, but can we do better? In particular, is it possible to answer such queries with a number of operations that is *sublinear* in $n = |V|$ or possibly even independent of n?

This local version of the problem was first studied in [8]. Observe that the main challenge is to answer queries about different edges consistently with the same sparse spanning graph G', while being allowed to inspect only a small (local) part of the graph for each queried edge. Moreover, while the underlying spanning graph may depend on the internal randomness of the algorithm that answers the queries, it should not depend on the (order of the) queries themselves. The following simple but important observation appears in [8]. If one insists that G have a minimum number of edges, namely, that G' be a tree, then it is easy to see that there are graphs G that contain edges e for which determining whether $e \in G'$ requires inspecting a linear number of edges in G. To see this, observe that if G consists of a single path, then the algorithm must answer positively on all edges, while if G consists of a cycle, then the algorithm must answer negatively on one edge. However, the two cases cannot be distinguished without inspecting a linear number of edges.

Given the above observation, the question posed in [8] is whether a relaxed version of this task can be solved by inspecting a sublinear number of edges, where the relaxation is that the spanning graph G' may contain $(1 + \epsilon) \cdot n$ edges, for a given approximation/sparsity parameter ϵ. As shown in [8], in general, even after this relaxation, local algorithms for sparse subgraphs require the inspection of $\Omega(\sqrt{n})$ edges for each queried edge e and for a constant ϵ. This lower bound holds for graphs with strong expansion properties, and there is an almost matching upper bound for such graphs. In fact, even for graphs with relatively weak expansion properties, there is no local algorithm that inspects a number of edges that is independent of n [9]. However, for graphs that are sufficiently non-expanding there are local algorithms whose complexity is independent of n (depending only on the maximum degree d and the approximation parameter ϵ) [8, 9].

A family of non-expanding graphs that is of wide interest, is the family of *minor-free* graphs, which can be parameterized by the size, h, of the excluded minor. In particular, this family includes planar graphs. It was shown in [8] that, based on [7], it is possible to obtain a local sparse spanning tree algorithm for graphs that are free of a minor of size h, where the complexity of the algorithm is $(d/\epsilon)^{\mathrm{poly}(h)\log(1/\epsilon)}$ and which has high constant success probability.

1.1 Our Result and Techniques

In this work we significantly improve on the aforementioned result for the class of minor-free graphs, by designing a local sparse spanning graph algorithm for minor-free graphs whose complexity is polynomial in h, d and $1/\epsilon$ and which has success probability $1 - 1/\Omega(n)$. We note that though graphs with excluded minors have bounded average degree, the *maximum* degree of such graphs remains unbounded, and our local algorithms have complexity that depends on this maximum degree. The spanning graph G' maintains the minimum distances between pairs of vertices in G up to a factor of $\tilde{O}(h \log(d)/\epsilon)$. Namely, it has *stretch* [14, 13] $\tilde{O}(h \log(d)/\epsilon)$ (the stretch obtained in [8] is somewhat higher, and in particular polynomial in d).

Similarly to some of the other local algorithms for sparse spanning graphs presented in [8, 9], our algorithm is based on defining an underlying global partition (which is randomized). The global partition determines the sparse spanning graph, and the local algorithm decides whether a given edge belongs to the spanning graph by constructing parts of the partition. The differences between the algorithms are in the way the partition is defined, the way the spanning graph is determined by the partition (more specifically, the choice of edges between parts), and the local partial construction of the partition. Interestingly, our partition and its construction bear more similarities to the underlying partition defined by the local sparse spanning graph algorithm for highly expanding graphs, than the partition used by the previous algorithm for minor-free graphs [8]. We discuss the similarities and differences further in Subsection 1.1.8. In what follows we provide a high level description of the partition, the sparse spanning graph it defines, and the corresponding local algorithm. We also explain how we can directly obtain a distributed algorithm, and give a bound on its round complexity.

1.1.1 A partition-based algorithm and the construction of G'

Our local algorithm is based on defining an underlying global partition of the vertices into many small connected parts, where the partition satisfies certain properties defined in the next paragraph. Given such a partition, the edge set E' of G' is simply defined by taking a spanning tree in each part, and a single edge between every pair of parts that have at least one edge between them. The minor-freeness of the graph, together with a bound on the number of parts, ensure that $|E'| \leq (1+\epsilon)n$.

1.1.2 Properties of the partition

We define a partition that has each of the four following properties (which for simplicity are not precisely quantified in this introductory text): (1) the number of parts in the partition is not too large; (2) the size of each part is not too large; (3) each part induces a connected subgraph; (4) for every vertex v it is possibly to efficiently find the subset of vertices in the part that v belongs to.

1.1.3 An initial centers-based partition

Initially, the partition is defined by a selection of *centers*, where the centers are selected randomly (though not completely independently). Each vertex is initially assigned to the closest selected center. For an appropriate setting of the probability that a vertex is selected to be a center, with high probability, this initial partition has the first property. Namely, the number of parts in the partition is not too large. By the definition of the partition, each part is connected, so that the partition has the third property as well. However, the partition is not expected to have the second property, that of each part having small size. Even for a simple graph such as the line, we expect to see parts of size logarithmic in n. In addition, the same example shows that the fourth property may not hold, i.e. there may be many vertices for which it takes superconstant time to find the part that the vertex belongs to. To deal with these two problems, we give a procedure for refining the partition in two phases, as explained next.

1.1.4 Refining the partition, phase 1 (and a structural lemma)

A first refinement is obtained by the separation of vertices that in order to reach their assigned center in a Breadth First Search (BFS), must observe a number of vertices that is above a certain predetermined threshold, k. Each of these *remote* vertices becomes a singleton subset in the partition. We prove that with probability $1 - 1/\Omega(n)$, the number of these remote vertices is not too large, so that the first property (concerning the number of parts) is maintained with high probability.

The probabilistic analysis builds on a structural lemma, which may be of independent interest. The lemma upper bounds, for any given vertex v, the number of vertices u such that v can be reached from u by performing a BFS until at most k vertices are observed. This number in turn upper bounds the size of each part in the partition after the aforementioned refinement. While this upper bound is not polynomial in k and d, it suffices for the purposes of our probabilistic (variance) analysis (and we also show that there is no polynomial upper bound).

1.1.5 Refining the partition, phase 2

In addition to the first property, the third property (connectivity of parts) is also maintained in the resulting refined partition, and the refinement partially addresses the fourth property, as remote vertices can quickly determine that they are far from all centers. In addition, the new parts of the partition will be of size 1 and thus will not violate the second property. However, after this refinement, there might be some large parts remaining so that the second and fourth properties are not necessarily satisfied. We further partition large parts into smaller (connected) parts by a certain decomposition based on the BFS-tree of the large part.

1.1.6 The local algorithm

Given an edge $\{u, v\} \in E$, the main task of the local algorithm is to find the two parts to which the vertices belong in the final partition. Once these parts are found, the algorithm can easily decide whether the edge between u and v should belong to E' or not. In order to find the part that u (similarly, v) belongs to, the local algorithm does the following. First it performs a BFS until it finds a center, or it sees at least k vertices (none of which is a center). In the latter case, u is a singleton (remote) vertex. In the former case, the algorithm has found the center that u is assigned to in the initial partition, which we denote by $\sigma(u)$. The algorithm next performs a BFS starting from $\sigma(u)$. For each vertex that it encounters, it checks whether this vertex is assigned to u (in the initial partition). If the number of vertices that are found to be assigned to $\sigma(u)$ is above a certain threshold, then the decomposition of the part assigned to $\sigma(u)$ needs to be emulated locally. We show that this can be done recursively in an efficient manner.

1.1.7 A distributed algorithm

Our algorithm easily lends itself to an implementation in the distributed (message passing) setting. In this setting a processor resided on each vertex in the graph. Computation proceeds in rounds, where in each round every vertex sends messages to its neighbors. In the distributed implementation, initially each vertex decided, independently at random (and with the appropriate probability), whether it is a center (of the initial partition). In the first round, each vertex sends its id (name) to each of its neighbors, as well as an indication whether it is a center. In the following rounds, each center send all its neighbors the information it has gathered about its local neighborhood (including the identity of the centers). It follows from our analysis that after $\tilde{O}(h \log(d)/\epsilon))$ rounds, each processor can determine if its incident edges belong to the sparse spanning graph G'.

1.1.8 A discussion of the algorithm for highly expanding graphs in [8]

As noted previously, there are similarities between our algorithm and the algorithm described in [8] for highly expanding graphs (for which the lower bound of $\Omega(\sqrt{n})$ holds). We refer to the latter algorithm as Centers-LRR, and provide a rough description of it next. The Centers-LRR algorithm is based on the selection of a set of random centers, which induces a partition of all the vertices, where each vertex is assigned to the center it is closest to. The number of centers is $\sqrt{\epsilon n}$, so that if we take the edges of a BFS tree for each part, and at most one edge between each pair of parts, we get at most $(1+\epsilon)n$ edges.

Based on the assumption regarding the expansion properties of the graph, with high probability over the random choice of the centers, for an appropriate threshold k, the distance between every vertex and its center is at most k, and the size of the distance-k neighborhood of every vertex is $\tilde{O}(\sqrt{n/\epsilon})$. This implies that each vertex can find its center by performing these many queries (for a constant degree). Hence, for an edge between two vertices that are assigned to the same center, we can easily determine whether it is a BFS edge (which belongs to the sparse spanning subgraph).

On the other hand, the decision regarding edges between vertices that are assigned to different centers is more subtle. Since at most one edge between every pair of parts is allowed, it seems that it is necessary to determine, for a given center, the identity of all vertices that are assigned to it. Though the number of such vertices is not too large, it is not clear how to perform this task in a local and efficient manner. The Centers-LRR algorithm overcomes this obstacle by defining a rule for the selection of at most one edge between every pair of parts, which can be implemented locally and efficiently. This rule is such that for some pairs of parts that have edges between them, it might be that no edge is selected. Nonetheless, it is proved in [8] that the final resulting subgraph is connected.

If we compare the Centers-LRR algorithm to the algorithm presented in this work we see that while there is a clear similarity in terms of the underlying partition (which is only initial in the current work), there are many differences in the difficulties that need to be overcome and the local implementation.

1.2 Related Work

As mentioned earlier, the works most closely related to the current work are [8, 9]. In addition to the results in these works that were already described, in [8] there is a local sparse spanning graph algorithm for the family of ρ-hyperfinite graphs[1] whose time complexity and probe complexity are $O(d^{\rho(\epsilon)})$, assuming that ρ is known. Minor-free graphs are a subclass of hyperfinite graphs (for an appropriate choice of ρ that depends on the size of the excluded minor). In [9] the authors show that in every f-non-expanding graph[2] G where $f(t) = \Omega((\log t)^{-1}(\log\log t)^{-2})$ one can remove ϵn edges from G so that each connected component of the remaining graph is of size at most $2^{2^{O(1/\epsilon)}}$. This enables them to provide a local sparse spanning graph algorithm for this family of graphs with probe complexity $d^{2^{2^{O(1/\epsilon)}}}$ and stretch $2^{2^{O(1/\epsilon)}}$. Both algorithms, in [8] and in [9], sparsify the graph by simulating a localized version of Kruskal's algorithm.

[1] A graph is *ρ-hyperfinite* for a function $\rho : \mathbb{R}_+ \to \mathbb{R}_+$, if its vertices can be partitioned into subsets of size at most $\rho(\epsilon)$ that are connected and such that the number of edges between the subsets is at most ϵn.

[2] A graph is *f-non-expanding* if every t-vertex subgraph H satisfies $\phi_H \le f(t)$ where ϕ_G is the (edge) *expansion* of G, that is, $\phi_G = \min_S |\partial_G(S)| / |S|$ where the minimum is taken over all $S \subseteq V(G)$ of size $1 \le |S| \le |V(G)|/2$.

By using the partition oracle in [7] one can obtain a local sparse spanning graph algorithm with quasi-polynomial complexity in d and ϵ^{-1}. The partition is obtained by contracting edges iteratively in a controlled manner. The advantage of their partition is that its edge cut is guaranteed to be small (with high constant probability). However, as mentioned before, their complexity is much higher.

Graph partitioning for graphs with an excluded minor has been studied extensively (see e.g., [3, 6]), and was found useful in constructing spanners and distance oracles (see e.g., [5]). However, compared to previous results, the main benefit of our partitioning technique is that the partition is designed to be constructed locally in an efficient manner.

Ram and Vicari [15] studied the problem of constructing sparse spanning graphs in the distributed model and provided an algorithm that runs in $\min\{D(G), O(\log n)\}$ number of rounds where $D(G)$ denotes the diameter of G.

The model of *local computation algorithms* as considered in this work, was defined by Rubinfeld et al. [16] (see also Alon et al. [2]). Other local algorithms, for maximal independent set, hypergraph coloring, k-CNF and maximum matching include those given in [16, 2, 11, 12, 4].

2 Preliminaries

In this section we provide the precise definition of the algorithmic problem addressed in this paper, as well as several other useful definitions and notations.

We consider undirected, simple graphs over n vertices, where the degree of each vertex is bounded by d. We assume for simplicity that the set of vertices, V, is simply $[n] = \{1, \ldots, n\}$, so that there is a total order over the (identifiers of the) vertices. For each vertex v, there is some arbitrary, but fixed order over its neighbours. The input graph $G = (V, E)$ is given via an *oracle access to its incidence-list representation*. Namely, the algorithm is supplied with n and d, and has access to an oracle that for any pair (v, i) such that $v \in [n]$ and $i \in [d]$, the oracle either returns the i^{th} neighbour of v (according to the aforementioned order over neighbours) or an indication that v has less than i neighbours. We refer to each such access to the oracle as a *probe* to the graph. We now turn to formally define the algorithmic problem we consider in this paper.

▶ **Definition 1.** An algorithm $\mathcal{A}$ is an (ϵ, q, δ)-*local sparse spanning graph algorithm* if, given $n, d \geq 1$ and oracle access to the incidence-lists representation of a connected graph $G = (V, E)$ over n vertices and degree at most d, it provides query access to a subgraph $G' = (V, E')$ of G such that:

(i) G' is connected.
(ii) $|E'| < (1 + \epsilon) \cdot n$ with probability at least $1 - \delta$ (over the internal randomness of $\mathcal{A}$).
(iii) E' is determined by G and the internal randomness of $\mathcal{A}$.
(iv) $\mathcal{A}$ makes at most q probes to G.

By "providing query access to G'" we mean that on input $\{u, v\} \in E$, $\mathcal{A}$ returns whether $\{u, v)\} \in E'$ and for any sequence of queries, $\mathcal{A}$ answers consistently with the same G'.

An algorithm $\mathcal{A}$ is an (ϵ, q, δ)-*local sparse spanning graph algorithm for a family of graphs* $\mathcal{C}$ if the above conditions hold, provided that the input graph G belongs to $\mathcal{C}$.

We are interested in local algorithms that for each edge they are queried on, perform as few probes as possible to G. Ideally, we would like the number of probes to be independent of n and polynomial in $1/\epsilon$, d, and possibly some parameters of the family $\mathcal{C}$. We are also interested in bounding the total amount of space used by the local algorithm, and its running time (in the word-RAM model). Note that Item 3 implies that the answers of the algorithm to queries cannot depend on previously asked queries.

We denote by $\text{dist}_G(u, v)$ (and sometimes by $\text{dist}(u, v)$ when G is clear from the context) the distance between two vertices u and v in G. We let $N(v)$ denote the set of neighbors of v and for $\ell \geq 0$ let $\Gamma_\ell(v) \overset{\text{def}}{=} \{u \in V : \text{dist}(u, v) \leq \ell)\}$.

Another parameter of interest is the stretch of G'. Given a connected graph $G = (V, E)$, a subgraph $G' = (V, E')$ is a t -spanner of G if for every $u, v \in V$, $\frac{\text{dist}_{G'}(u,v)}{\text{dist}_G(u,v)} \leq t$. In this case t is referred to as the *stretch factor* of G'.

The total order over the vertices induces a total order (ranking) r over the edges of the graph in the following straightforward manner: $r(\{u, v\}) < r(\{u', v'\})$ if and only if $\min\{u, v\} < \min\{u', v'\}$ or $\min\{u, v\} = \min\{u', v'\}$ and $\max\{u, v\} < \max\{u', v'\}$ (recall that $V = [n]$). The total order over the vertices also induces an order over those vertices visited by a Breadth First Search (BFS) starting from any given vertex v, and whenever we refer to a BFS, we mean that it is performed according to this order.

Recall that a graph H is called a *minor* of a graph G if H is isomorphic to a graph that can be obtained by zero or more edge contractions on a subgraph of G. A graph G is H*-minor free* if H is not a minor of G. We will use the following theorem.

▶ **Theorem 2** ([10]). *Let $c(s)$ be the minimum number c such that every graph $G = (V, E)$ with $|E| \geq c \cdot |V|$ contracts to a complete graph K_s. Then $c(s) \leq 8s \log s$.*

We shall use the following result from previous work (see Section 6 in [1]).

▶ **Theorem 3.** *For every $1 \leq t \leq n$, there exists an explicit construction of a t-wise independent random variable $x = (x_1, \ldots, x_n) \in [q]^n$ for $q = \Theta(n)$ whose seed length is at most $O(t \log n)$ bits. Moreover, for any $1 \leq i \leq n$, x_i can be computed in $O(t)$ operations in the word-RAM model.*

3 The Algorithm

In this section we prove the following theorem.

▶ **Theorem 4.** *Algorithm 2 is an $(\epsilon, \text{poly}(1/\epsilon, d, h), \delta)$-local sparse spanning graph algorithm for graphs that are H-minor free, where h is the size of H and $\delta = 1/\Omega(n)$. Furthermore, the stretch factor of G' is $\tilde{O}(h \cdot \log d/\epsilon)$. More precisely, the probe complexity of the algorithm is $\tilde{O}((h/\epsilon)^4 d^5)$. Its space complexity (length of the random seed) is $\tilde{O}((h/\epsilon) d \log n)$ bits, and its running time is $\tilde{O}((h/\epsilon)^5 d^5)$ (in the Word-RAM model).*

We begin by describing a global partition of the vertices. We later describe how to locally generate this partition and design our algorithm (and the sparse subgraph it defines), based on this partition.

3.1 The Partition $\mathcal{P}$

The partition described in this subsection, denoted by $\mathcal{P}$, is a result of a random process. We describe how the partition is obtained in three steps where in each step we refine the partition from the previous step. The partition is defined based on three parameters: $\gamma \in (0, 1)$, an integer $k > 1$ and an integer $s > 1$, which is set subsequently (as polynomials of d, ϵ and h).

3.1.1 First Step

We begin with some notation. Given a subset of vertices $W \subseteq V$ and a vertex $v \in V$, we define the *center* of v with respect to W, denoted $\sigma(v)$ as the vertex in W that is closest to v, breaking ties using vertex ids. That is, $\sigma(v)$ is the vertex with the minimum identifier in the subset $\{y \in W : \text{dist}(y, v) = \min_{w \in W} \text{dist}(w, v)\}$.

For each $w \in W$ we define the *cell* of w with respect to W as $C(w) \overset{\text{def}}{=} \{v \in V : \sigma(v) = w\}$. Namely, the set of vertices in $C(w)$ are the vertices which are closer to w more than any other vertex in W (where ties are broken according to the order of the vertices). Notice that these cells form a partition of V. Our initial partition is composed of these cells when picking W in the following way: each vertex $i \in [n]$ draws a γ-biased bit, denoted x_i, and $i \in W$ if an only if $x_i = 1$. The joint-distribution of $(x_1, \ldots, x_n)$ is t-wise independent where $t \overset{\text{def}}{=} 2kd$. (The reason that the choice of W is determined by a t-wise independent distribution rather than an n-wise independent distribution is so as to bound the space used by the local emulation of the global algorithm.)

3.1.2 Second Step

In this step we identify a subset of *special* vertices, which we call the *remote vertices* and make these vertices singletons in the partition $\mathcal{P}$. The set of remote vertices, R, is defined with respect to W and an integer parameter k as described next. Let $\ell_k(v)$ be the minimum integer ℓ such that the BFS tree rooted at v of depth ℓ has size at least k. Let $B_k(v)$ be the set of vertices in the BFS tree rooted at v of depth $\ell_k(v)$. We define $R = \{v \in V : B_k(v) \cap W = \emptyset\}$, i.e., those vertices v for which $B_k(v)$ does not contain a center. Clearly, a vertex can identify efficiently if it is in R by probing at most kd vertices and checking whether they intersect W. In Subsection 3.3 we obtain a bound on the size of R.

3.1.3 Third Step

In this step we decompose cells that are still too big. We first argue that the cells are still connected (after the removal the vertices in R from all original cells). Thereafter we will use a procedure of tree decomposition in order to break the cells into smaller parts.

▶ **Lemma 5.** *For every $w \in W$, the subgraph induced by $C(w) \setminus R$ is connected. Furthermore, for every $v \in C(w) \setminus R$, the subgraph induced by $C(w) \setminus R$ contains all vertices that belong to the shortest paths between v and w.*

Proof. Fix $w \in W$, and consider any $v \in C(w) \setminus R$. We will prove that the subgraph induced by $C(w) \setminus R$ contains all vertices on the shortest paths between v and w and this will imply the connectivity as well. The proof is by induction on the distance to w. In the base case $v = w$. In this case $C(w) \setminus R$ clearly contains a path between v to itself because it contains v. Otherwise, we would like to show that for any $u \in N(v)$ for which $\text{dist}(u, w) < \text{dist}(v, w)$ it holds that $u \in C(w) \setminus R$. The proof will then follow by induction. let P be a shortest path between v and w and let $\{v, u\} \in E$ denote the first edge in P. We first observe that $\sigma(u) = w$ and thus $u \in C(w)$. Assume otherwise and conclude that there is a vertex in $w' \in W$ for which either $\text{dist}(v, w') < \text{dist}(v, w)$ or $\text{dist}(v, w) = \text{dist}(v, w')$ and $id(w') < id(w)$, in contradiction to the fact that $\sigma(v) = w$ (see the definition of $\sigma(\cdot)$ in the First Step). Since u is on a shortest path between v and w it follows that

$$\Gamma_{\text{dist}(u,w)-1}(u) \subseteq \Gamma_{\text{dist}(v,w)-1}(v) \ . \qquad (1)$$

From the fact that $v \notin R$ it follows that $|\Gamma_{\mathrm{dist}(v,w)-1}(v)| \leq k$ and hence from Equation (1) it follows that $|\Gamma_{\mathrm{dist}(u,w)-1}(u)| \leq k$ and so $u \notin R$ as well. We conclude that $u \in C(w) \setminus R$ and $\mathrm{dist}(u,w) = \mathrm{dist}(v,w) - 1$ as desired. ◀

We shall use the following notation. For each $w \in W$ let $\mathcal{T}(w)$ denote the BFS tree rooted at w of the subgraph induced by $C(w) \setminus R$ (recall that the BFS is performed by exploring the vertices according to the order of their identifiers (in $[n]$)). To obtain the final refinement of our partition, for each $w \in W$ such that $|\mathcal{T}(w)| > s$, we run Algorithm 1 on $\mathcal{T}(w)$, w and s.

Algorithm 1 (Recursive Tree decomposition)

```
Input: A tree $\mathcal{T}$, the root of the tree $v$ and an integer $s$.
Output: A decomposition of $\mathcal{T}$ into subtrees, where each subtree is assigned a (sub-)center.
1. Initialize the set of vertices of the current part $Q := \emptyset$.
2. Perform a BFS starting from $v$ and stop at level $\ell \overset{\mathrm{def}}{=} \ell_s(v)$ (see the definition of $\ell_s(\cdot)$ in the
   Second Step). Add to $Q$ all the vertices explored in the BFS.
3. Let $S$ denote the set of all the children of the vertices in the $\ell^{\mathrm{th}}$ level of the BFS (namely, all the
   vertices in level $\ell + 1$).
4. For each vertex $u \in S$:
   a. If the subtree rooted at $u$, $\mathcal{T}_u$, has size at least $s$, then disconnect this subtree from $\mathcal{T}$ and
      continue to decompose by recursing on input $\mathcal{T}_u$, $u$ and $s$.
   b. Otherwise, add the vertices of $\mathcal{T}_u$ to $Q$.
5. Set $v$ to be the sub-center of all the vertices in $Q$.
```

3.2 The Edge Set

Given the partition $\mathcal{P}$ defined in the previous subsection (Subsection 3.1), we define the edge set of our sparse spanning graph E' in the following simple manner. In each part of $\mathcal{P}$ which is not a singleton, we take a spanning tree. Specifically, we take the BFS-tree rooted at the sub-center of that part (see Algorithm 1, Step 5). For every pair of parts of $X, Y \in \mathcal{P}$, if the set $E(X,Y) \overset{\mathrm{def}}{=} \{\{x,y\} \in E : x \in X \text{ and } y \in Y\}$ is not empty, then we add to E' the edge $e \in E(X,Y)$ with minimal ranking (where the ranking over edges is as defined in Section 2). Clearly G' is connected and spans G. We would like to bound the size of E'. To this end we will use the following claim.

▶ **Claim 6.** *The number of parts in $\mathcal{P}$ is bounded as follows: $|\mathcal{P}| \leq |W| + |R| + \frac{n}{s}$.*

Proof. Consider the three steps of refinement of the partition. Clearly, after the first step the size of the partition is exactly $|W|$. After the second step, the size of the partition is exactly $|W| + |R|$. Finally, since in the last step we break only parts whose size is greater than s into smaller parts that are of size at least s, we obtain that the number of new parts that are introduced in this step is at most n/s. The claim follows. ◀

The next lemma establishes the connection between the size of $\mathcal{P}$ and the sparsity of G'.

▶ **Lemma 7.** *For an input graph G which is a H-minor free for a graph H over h vertices,*

$$|E'| < n + |\mathcal{P}| \cdot c(h) ,$$

where $c(h)$ is as defined in Theorem 2.

Proof. Since for each $X \in \mathcal{P}$ the subgraph induced by X is connected, we can contract each part in $\mathcal{P}$ and obtain an H-minor free graph. The number of vertices in this graph is $|\mathcal{P}|$. If we replace multi-edges with single edges, then by Theorem 2 we obtain that the number of edges in this graph is at most $|\mathcal{P}| \cdot c(h)$. Finally, since the total number of edges in the union of spanning trees of each part it $n - |\mathcal{P}| < n$, we obtain the desired result. ◀

3.3 Bounding the Number of Remote Vertices

In this subsection we prove the following lemma.

▶ **Lemma 8.** *If $k = \Omega((\log^2(1/\gamma) + \log d)/\gamma)$, then with probability at least $1 - \frac{1}{\Omega(n)}$ it holds that $|R| \le \gamma n$.*

In order to establish Lemma 8 we start defining for every $v \in V$,

$$Y_k(v) \stackrel{\text{def}}{=} \{u \in V : v \in B_k(u)\} \ . \tag{2}$$

Informally, $Y_k(v)$ is the set of vertices that encounter v while performing a BFS which stops after the first level in which the total number of explored vertices is at least k. We first establish the following simple claim.

▶ **Claim 9.** *For every vertex $u \in Y_k(v)$ and for every vertex w that is on a shortest path between u and v, we have that $w \in Y_k(v)$.*

Proof. Let $\ell = \text{dist}(u, v)$ and $\ell' = \text{dist}(w, v)$, so that $d(u, w) = \ell - \ell' \ge 1$. Assume, contrary to the claim, that $w \notin Y_k(v)$. This implies that $|\Gamma_{\ell'-1}(w)| \ge k$. But since $\Gamma_{\ell'-1}(w) \subseteq \Gamma_{\ell-1}(u)$, we get that $|\Gamma_{\ell-1}(u)| \ge k$, contrary to the premise of the claim that $u \in Y_k(v)$. ◀

We now turn to upper bound the size of $Y_k(v)$.

▶ **Lemma 10.** *For every graph $G = (V, E)$ with degree bounded by d, and for every $v \in V$,*

$$|Y_k(v)| \le d^3 \cdot k^{\log k + 1} \ .$$

Proof. Fix a vertex $v \in V$. For every $0 \le j \le k$, define $Y_k^j(v) \stackrel{\text{def}}{=} \{u \in Y_k(v) : \text{dist}(v, u) = j\}$. Observe that $Y_k(v) = \bigcup_{j=0}^{k} Y_k^j(v)$. Therefore, if we bound the size of $Y_k^j(v)$, for every $0 \le j \le k$, we will get a bound on the size of $Y_k(v)$. Consider first any $3 \le j < k$ and any vertex $u \in Y_k^j(v)$. Recall that $\ell_k(u)$ is the the minimum integer ℓ such that the BFS tree rooted at v of depth ℓ has size at least k. Since $j \le \ell_k(u)$, it follows that $|\Gamma_{j-1}(u)| < k$. Now consider a shortest path between u and v and let w be the vertex on this path for which $\text{dist}(u, w) = \lfloor (j-1)/2 \rfloor$. Denote $q \stackrel{\text{def}}{=} \text{dist}(w, v)$. By Claim 9, $w \in Y_k(v)$, and by the definition of q, $w \in Y_k^q(v)$. Therefore,

$$|\Gamma_{q-1}(w)| \le k \ . \tag{3}$$

From the fact that w is on the shortest path between u and v it also follows that

$$\begin{aligned} q &= \text{dist}(v, u) - \text{dist}(u, w) = j - \lfloor (j-1)/2 \rfloor \\ &= \lceil (j-1)/2 \rceil + 1 \\ &\ge \lfloor (j-1)/2 \rfloor + 1 = \text{dist}(u, w) + 1 \ . \end{aligned} \tag{4}$$

Therefore $q - 1 \ge \text{dist}(u, w)$ and so $u \in \Gamma_{q-1}(w)$. It follows that

$$Y_k^j(v) \subseteq \bigcup_{w \in Y_k^q(v)} \Gamma_{q-1}(w) \ . \tag{5}$$

From Equations (3) and (5) we get that $|Y_k^j(v)| \le k \cdot |Y_k^q(v)|$. For every $j \le 3$ we have the trivial bound that $|Y_k^j(v)| \le d^3$. By combining with Equation (4) we get that $|Y_k^j(v)| \le d^3 \cdot k^{\log j}$. Since $Y_k(v) = \bigcup_{j=0}^{k} Y_k^j(v)$ we obtain the desired bound. ◀

While the bound on $|Y_k(v)|$ in Lemma 10 may not be tight, it suffices for our purposes. One might conjecture that it is possible to prove a polynomial bound (in k and d). We show that this is not the case (see Lemma 11 in the appendix).

We now use the bound in Lemma 10 in order to bound the number of remote vertices.

Proof of Lemma 8. Let χ_W denote the characteristic vector of the set W. For a subset $S \subseteq V$, let $\chi_W(S)$ denote the projection of χ_W onto S. That is, $\chi_W(S)$ is a vector of length $|S|$ indicating for each $x \in S$ whether $\chi_W(x) = 1$.

For each vertex $v \in V$ define a random variable Z_v indicating whether it is a remote vertex with respect to W. Recall that v is remote if and only if $B_k(v) \cap W = \emptyset$. Recalling that W is selected according to a t-wise independent distribution where $t = 2kd$ and that $k \le |B_k(v)| < k \cdot d$, we get that $\Pr[Z_v = 1] \le (1-\gamma)^k$. We also define $S_v \overset{\text{def}}{=} \{u \in V : B_k(u) \cap B_k(v) \ne \emptyset\}$. Fix $v \in V$ and observe that the value of Z_v is determined by $\chi_W(B_k(v))$. Furthermore, since for every $v \in V$ and $u \in V \setminus S_v$, $\chi_W(B_k(v))$ and $\chi_W(B_k(u))$ are independent it follows that Z_u and Z_v are independent as well. Hence, in this case $\mathrm{Cov}[Z_v, Z_u] = 0$, and we obtain the following upper bound on the variance of the number of remote vertices.

$$\begin{aligned}\mathrm{Var}\left[\sum_{v\in V} Z_v\right] &= \sum_{(v,u)\in V} \mathrm{Cov}[Z_v, Z_u] = \sum_{v\in V}\sum_{u\in S_v} (\mathrm{Exp}[Z_v\cdot Z_u] - \mathrm{Exp}[Z_u]\cdot\mathrm{Exp}[Z_v]) \\ &\le \sum_{v\in V}\sum_{u\in S_v} \mathrm{Exp}[Z_v \cdot Z_u | Z_v = 1]\cdot \Pr[Z_v = 1] \le \sum_{v\in V} |S_v|\cdot(1-\gamma)^k \ . \quad (6)\end{aligned}$$

By the definition of $Y_k(\cdot)$ in Equation (2) it follows that $S_v \subseteq \bigcup_{u\in B_k(v)} Y_k(u)$. By Lemma 10, $\max_{v\in V}\{|Y_k(v)|\} \le d^3 \cdot k^{\log k+1}$. Therefore

$$|S_v| \le |B_k(v)| \cdot d^3 \cdot k^{\log k+1} \le d^4 \cdot k^{\log k+2} \ . \quad (7)$$

Hence, by Equations (6) and (7) we get $\mathrm{Var}\left[\sum_{v\in V} Z_v\right] \le nd^4 \cdot k^{\log k+2}\cdot(1-\gamma)^k$. Since $(1-\gamma)^k \le e^{-\gamma k}$ we obtain that $\mathrm{Var}\left[\sum_{v\in V} Z_v\right] \le \gamma^2 n$ for $k = \Omega((\log^2(1/\gamma) + \log d)/\gamma)$. Since for every $v \in V$, $\Pr[Z_v = 1] \le (1-\gamma)^k \le \gamma$, we get that $\mathrm{Exp}\left[\sum_{v\in V} Z_v\right] \le \gamma n/2$. By applying Chebyshev's inequality we get that

$$\Pr\left[\sum_{v\in V} Z_v \ge \mathrm{Exp}\left[\sum_{v\in V} Z_v\right] + \gamma n/2\right] \le \frac{4\mathrm{Var}\left[\sum_{v\in V} Z_v\right]}{\gamma^2 n^2} \le \frac{4}{n} \ .$$

Since $|R| = \sum_{v\in V} Z_v$ it follows that $|R| < \gamma n$ with probability at least $1 - (4/n)$, as desired. ◀

3.4 The Local Algorithm

In this subsection we provide Algorithm 2, which on input $e \in E$, locally decides whether $e \in E'$, as defined in Subsection 3.2, based on the (random, but not completely independent) choice of W. Given an edge $\{u, v\}$, the algorithm first finds, for each $y \in \{u, v\}$, the center, $\sigma(y)$, that y is assigned to by the initial partition, under the condition that $\sigma(y) \in B_k(y)$. This is done by calling Algorithm 3, which simply performs a BFS starting from y until it encounters a vertex in W, or it reaches level $\ell_k(y)$ without encountering such a vertex (in which case y is a remote vertex). Algorithm 3 assumes access to χ_W, which is implemented using the random seed that defines the t-wise independent distribution, and hence determines W. If y is not found to be a remote vertex, then Algorithm 2 next determines to which sub-part of $C(\sigma(y)) \setminus R$ does y belong. This is done by emulating the tree decomposition of the BFS tree rooted at $\sigma(y)$ and induced by $C(\sigma(y)) \setminus R$. A central procedure that is employed in this process is Algorithm 4. This algorithm is given a vertex $v \in W$, and a vertex u in the BFS subtree rooted at v and induced by $C(v) \setminus R$. It returns the edges going from u to its children in this tree, by performing calls to Algorithm 3.

Algorithm 2 (Sparse Spanning Graph)

```
Input: An edge $\{u, v\} \in E$.
Output: YES if $\{u, v\} \in E'$ and NO otherwise.
1. For each $y \in \{u, v\}$ find the part that $y$ belongs to as follows:
   a. Use Algorithm 3 to obtain $\sigma(y)$.
   b. If $\sigma(y)$ is 'null' then the part that $y$ belongs to is the singleton set $\{y\}$.
   c. Let $\mathcal{T}$ denote the BFS tree rooted at $\sigma(y)$ in the subgraph induced by $C(\sigma(y)) \setminus R$. By
      Lemma 5 every shortest path between $\sigma(y)$ and $v \in C(\sigma(y)) \setminus R$ is contained in the subgraph
      induced on $C(\sigma(y)) \setminus R$. Therefore the edges of $\mathcal{T}$ can be explored via Algorithm 4.
   d. Reveal the part (the subset of vertices) that $y$ belongs to in $\mathcal{P}$ (as defined in Subsection 3.1).
      Recall that the part of $y$ is the subtree of $\mathcal{T}$ that contains $y$ after running Algorithm 1 on
      input $\mathcal{T}$, $\sigma(y)$ and $s$. This part can be revealed locally as follows.
      i. Reveal the path between $\sigma(y)$ and $y$ in $\mathcal{T}$, denoted by $P(y)$. Since $P(y)$ is the shortest path
         between $y$ and $\sigma(y)$ with the lexicographically smallest order it can revealed by performing
         a BFS from $y$ until $\sigma(y)$ is encountered.
      ii. Run Algorithm 1 on $\mathcal{T}$ while recursing in Step 4a only on the subtrees in which the root is
          contained in $P(y)$.
2. If $u$ and $v$ are in the same part, then return YES iff the edge $\{u, v\}$ belongs to the BFS tree of
   that part.
3. Otherwise, return YES iff the edge $\{u, v\}$ is the edge with minimum rank connecting the two
   parts.
```

Proof of Theorem 4. We set $\gamma = \epsilon/2c(h)$, $k = \Theta((\log^2(1/\gamma) + \log d)/\gamma)$, and $s = c(h)/4\epsilon$. We start by bounding the size of W (with high probability). By the definition of W we have that $\text{Exp}[|W|] = \text{Exp}\left[\sum_{i\in[n]} \chi_W(i)\right] = \gamma n$. Since for every $1 \le i < j \le n$, $\chi_W(i)$ and $\chi_W(j)$ are pairwise-independent, we obtain that

$$\text{Var}\left[\sum_{i\in[n]} \chi_W(i)\right] = \sum_{i\in[n]} \text{Var}\left[\chi_W(i)\right] = \sum_{i\in[n]} \left(\text{Exp}\left[\chi_W^2(i)\right] - \text{Exp}\left[\chi_W(i)\right]^2\right) = n\gamma(1-\gamma) \ .$$

Therefore, by Chebyshev's inequality $\Pr\left[|W| \ge 2\gamma n\right] \le \frac{1-\gamma}{\gamma n}$. By Lemma 8 (and the setting of k), with probability $1 - 1/\Omega(n)$, $|R| \le \gamma n$. By Claim 6, Lemma 7 and the settings of γ and s, we get that $|E'| \le (1+\epsilon)n$ with probability $1 - 1/\Omega(n)$.

The claim about the stretch of G' follows from the fact that the diameter of every part of $\mathcal{P}$ is bounded by $2k$.

The number of vertices that Algorithm 3 inspects (for any vertex v it is called with) is at most kd. Since the degree of each vertex is bounded by d, its probe complexity is bounded by kd^2. Algorithm 3 makes at most kd accesses to χ_W, hence, by Theorem 3 its running time is bounded by $O(k^2d^2)$. The probe complexity of Algorithm 4 is upper bounded by the total probe complexity of at most d calls to Algorithm 3, plus d executions of a BFS until at most kd vertices are encountered (Steps 2b and 2c, which for simplicity we account for separately from Algorithm 3). A similar bound holds for the running time. Hence, the total probe complexity and running time of Algorithm 4 are $O(kd^3)$ and $O(k^2d^3)$, respectively.

The size of any subtree returned by Algorithm 1 is upper bounded by s^2d^2. To verify this, recall that at the end of Step 2 of Algorithm 1, at most sd vertices were explored. Hence, the number of vertices that are incident to the explored vertices is at most sd^2. Thus, due to Step 4a, the total number of vertices in each part is at most s^2d^2. Since Step 4a can be implemented locally by calling Algorithm 4 at most s times, we obtain that the total probe complexity and running time of revealing each part is at most $O(s^2kd^5)$ and $O(s^2k^2d^5)$, respectively. The probe complexity and running time of Algorithm 2 are dominated by those of Step 1d. Observe that in Step 1d at most $|P(y)|$ parts are revealed. Since $|P(y)| \leq k$, we obtain that the overall probe complexity and running time of Algorithm 2 are bounded by $O(s^2k^2d^5)$ and $O(s^2k^3d^5)$, respectively. By the settings of k and s we obtain the final result. ◀

Algorithm 3 (Find Center)

```
Input: A vertex $v$ and an integer $k$. Query access to $\chi_W$.
Output: $\sigma(v)$ if $\sigma(v) \in B_k(v)$ or 'null' otherwise.
1. Perform a BFS from $v$ until the first level that contains a vertex in $W$ or until at least $k$ vertices are
   reached. That is, defining $W_j \overset{\text{def}}{=} \Gamma_j(v) \cap W$, stop at level $\ell$ where $\ell \overset{\text{def}}{=} \min\{\ell_k(v), \min_j\{W_j \neq \emptyset\}\}$.
2. If $W_\ell = \emptyset$ then return 'null' ($v$ is remote).
3. Otherwise, return the vertex with the minimum id from $W_\ell$.
```

Algorithm 4 (get BFS outgoing-edges endpoints)

```
Input: $v \in W$ and a vertex $u \in \mathcal{T}(v)$ where $\mathcal{T}(v)$ denotes the BFS tree induced by $C(v) \setminus R$ and
rooted at $v$.
Output: The outgoing edges from $u$ in $\mathcal{T}(v)$ (the orientation of the edges is from the root to the
leaves).
1. Initialize $S := \{v\}$
2. For each $u' \in N(u)$, if the following three conditions hold, then add $u'$ to $S$:
   a. Algorithm 3 on input $u'$ returns $v$. Namely, $\sigma(u') = v$.
   b. $u$ is on a shortest path between $u'$ and $v$. Namely, $\text{dist}(u', v) = \text{dist}(u, v) + 1$.
   c. $u$ is the vetex with the minimum id among all vertices in $\{w \in N(u') : \text{dist}(w, v) =
      \text{dist}(u', v) - 1\}$.
3. Return $S$.
```

Acknowledgement. We would like to thank Oded Goldreich for his helpful suggestions.

References

1 N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986. doi:10.1016/0196-6774(86)90019-2.

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 Noga Alon, Paul D. Seymour, and Robin Thomas. A separator theorem for graphs with an excluded minor and its applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 293–299, 1990. doi:10.1145/100216.100254.

4 G. Even, M. Medina, and D. Ron. Deterministic stateless centralized local algorithms for bounded degree graphs. In *Algorithms – ESA 2014 – 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 394–405, 2014. doi:10.1007/978-3-662-44777-2_33.

5 Ken-ichi Kawarabayashi, Philip N. Klein, and Christian Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *Automata, Languages and Programming – 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, pages 135–146, 2011. doi:10.1007/978-3-642-22006-7_12.

6 Ken-ichi Kawarabayashi and Bruce A. Reed. A separator theorem in minor-closed classes. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 153–162, 2010. doi:10.1109/FOCS.2010.22.

7 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.

8 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.

9 Reut Levi, Guy Moshkovitz, Dana Ron, Ronitt Rubinfeld, and Asaf Shapira. Constructing near spanning trees with few local inspections. *Random Structures & Algorithms*, pages n/a–n/a, 2016. doi:10.1002/rsa.20652.

10 W. Mader. Homomorphiesätze für graphen. *Mathematische Annalen*, 178:154–168, 1968.

11 Y. Mansour, A. Rubinstein, S. Vardi, and N. Xie. Converting online algorithms to local computation algorithms. In *Automata, Languages and Programming: Thirty-Ninth International Colloquium (ICALP)*, pages 653–664, 2012.

12 Y. Mansour and S. Vardi. A local computation approximation scheme to maximum matching. In *Proceedings of the Sixteenth International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 260–273, 2013.

13 D. Peleg and A. A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13:99–116, 1989.

14 D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM Journal on Computing*, 18:229–243, 1989.

15 L. S. Ram and E. Vicari. Distributed small connected spanning subgraph: Breaking the diameter bound. Technical report, Zürich, 2011.

16 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.

A A non-polynomial relation between k and $Y_k(v)$

Recall that $Y_k(v) \overset{\text{def}}{=} \{u \in V : v \in B_k(u)\}$. In the following lemma we show that $|Y_k(v)|$ can be super polynomial.

▶ **Lemma 11.** *There exists a graph $G = (V, E)$ with degree bounded by d and $v \in V$ such that $|Y_k(v)| = k^{\Omega(\log \log d)}$.*

Proof. The graph G is a tree, rooted at v, and defined as follows. For simplicity, we let the degree bound be $d + 1$ (so that a vertex may have d children, and hence degree $d + 1$). We partition the levels of the tree into consecutive subsets: $L_0 = \{1, \ldots, \ell_0\}$ (where the root is at level 1), $L_1 = \{\ell_0 + 1, \ldots, \ell_1\}$, $\ldots$, $L_r = \{\ell_{r-1} + 1, \ldots, \ell_r\}$. For each subset L_i, and for each level j in the subset, all vertices in level j have the same number of children, which is $d_i \overset{\text{def}}{=} d^{2^{-i}}$. We set $r = \log\log d$, so that all vertices in levels belonging to L_r have two children. Finally we set $s_i = |L_i| = \log_{d_i} g^{1/(r+1)}$, where g determines the size of the tree, as well as the minimum k that ensures that all vertices in the tree belong to $Y_k(v)$.

By the construction of the tree, the number of vertices in it is of the order of $\prod_{i=0}^{r} d_i^{s_i} = g$. In order to upper-bound k (such that all vertices belong to $Y_k(v)$), consider any vertex u in some level $j \in L_i$, where $0 \le i \le r$. Since $d_i = d_{i-1}^{1/2}$, so that $s_t = 2s_{t-1}$ for each t, we get that $s_i \le \sum_{i'<i} s_{i'}$. Therefore, $\text{dist}(u, v) \le 2s_i$. It follows that the number of vertices in the subtree rooted at u that are at distance at most $\text{dist}(u, v)$ from u is upper bounded by $d_i^{s_i} \cdot d_{i+1}^{s_i} < g^{3/2(r+1)}$. Since this is true for every vertex in the tree, we get that $|\Gamma_{\text{dist}(u,v)}(u)| = O(g^{3/2(r+1)} \cdot s_r) = O(g^{3/2(r+1)} \cdot \log g)$, which gives us an upper bound on k, from which the lemma follows. ◀

Tight Bounds for Sketching the Operator Norm, Schatten Norms, and Subspace Embeddings

Yi Li[*1] **and David P. Woodruff**[2]

1 **Facebook Inc., Seattle, WA, USA**
`leeyi@umich.edu`
2 **IBM Almaden Research Center, San Jose, CA, USA**
`dpwoodru@us.ibm.com`

Abstract

We consider the following oblivious sketching problem: given $\epsilon \in (0, 1/3)$ and $n \geq d/\epsilon^2$, design a distribution $\mathcal{D}$ over $\mathbb{R}^{k\times nd}$ and a function $f : \mathbb{R}^k \times \mathbb{R}^{nd} \to \mathbb{R}$, so that for any $n \times d$ matrix A,

$$\Pr_{S\sim\mathcal{D}}[(1-\epsilon)\|A\|_{op} \leq f(S(A), S) \leq (1+\epsilon)\|A\|_{op}] \geq 2/3,$$

where $\|A\|_{op} = \sup_{x:\|x\|_2=1} \|Ax\|_2$ is the operator norm of A and $S(A)$ denotes $S \cdot A$, interpreting A as a vector in $\mathbb{R}^{nd}$. We show a tight lower bound of $k = \Omega(d^2/\epsilon^2)$ for this problem. Previously, Nelson and Nguyen (ICALP, 2014) considered the problem of finding a distribution $\mathcal{D}$ over $\mathbb{R}^{k\times n}$ such that for any $n \times d$ matrix A,

$$\Pr_{S\sim\mathcal{D}}[\forall x,\ (1-\epsilon)\|Ax\|_2 \leq \|SAx\|_2 \leq (1+\epsilon)\|Ax\|_2] \geq 2/3,$$

which is called an oblivious subspace embedding (OSE). Our result considerably strengthens theirs, as it (1) applies only to estimating the operator norm, which can be estimated given any OSE, and (2) applies to distributions over general linear operators S which treat A as a vector and compute $S(A)$, rather than the restricted class of linear operators corresponding to matrix multiplication. Our technique also implies the first tight bounds for approximating the Schatten p-norm for even integers p via general linear sketches, improving the previous lower bound from $k = \Omega(n^{2-6/p})$ [Regev, 2014] to $k = \Omega(n^{2-4/p})$. Importantly, for sketching the operator norm up to a factor of α, where $\alpha - 1 = \Omega(1)$, we obtain a tight $k = \Omega(n^2/\alpha^4)$ bound, matching the upper bound of Andoni and Nguyen (SODA, 2013), and improving the previous $k = \Omega(n^2/\alpha^6)$ lower bound. Finally, we also obtain the first lower bounds for approximating Ky Fan norms.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases data streams, sketching, matrix norms, subspace embeddings

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.39

1 Introduction

Understanding the sketching complexity of estimating matrix norms [4, 14, 24, 29] has been a goal of recent work, generalizing a line of work on estimating frequency moments in the sketching model [3, 17, 23], and in the somewhat related streaming model of computation [1].

In the sketching model, one fixes a distribution $\mathcal{D}$ over $k \times (nd)$ matrices S, and is then given an $n \times d$ matrix A which, without loss of generality, satisfies $n \geq d$. One then samples

[*] Yi Li was supported by ONR grant N00014-14-1-0632 when he was at Harvard University, where the major part of this work was done.

S from $\mathcal{D}$, and computes $S(A)$, which denotes the operation of treating A as a column vector in $\mathbb{R}^{nd}$ and left-multiplying that vector by the matrix S. Any linear transformation applied to A can be expressed in this form, and therefore we sometimes refer to such a distribution $\mathcal{D}$ as a *general linear sketch*. There is also the related notion of a *bilinear sketch*, in which one fixes a distribution $\mathcal{D}$ over $k \times n$ matrices S, and is then given an $n \times d$ matrix A. One samples S from $\mathcal{D}$ and computes $S \cdot A$. Bilinear sketches are special cases of general linear sketches since they form a subclass of all possible linear transformations of A, and general linear sketches can be much more powerful than bilinear sketches. For example, to compute the trace exactly of an $n \times n$ matrix A, setting $k = 1$ suffices for a general linear sketch, while we do not know how to compute the trace with a small value k for bilinear sketches, and several lower bounds on k are known even to approximate the trace [28].

The goal in the sketching model is to minimize the *sketching dimension* k so that $S(A)$ can be used to approximate a property of A with constant probability. Associated with distribution $\mathcal{D}$ is an estimation procedure, which we model as a function f, for which $f(S(A), S)$ outputs a correct answer to the problem at hand with constant probability. For numerical properties, such as estimating a norm of A, this probability can be amplified to $1 - \delta$, by creating a distribution $\mathcal{D}'$ corresponding to taking $O(\log(1/\delta))$ independent copies $S^1, \ldots, S^{\log(1/\delta)}$ from $\mathcal{D}$, and outputting the median of

$$f(S^1(A), S^1), f(S^2(A), S^2), \ldots, f(S^{\log(1/\delta)}(A), S^{\log(1/\delta)}).$$

Notice that the mapping S is linear and oblivious, both of which are important for a number of applications such as merging sketches in distributed computation, or for approximately recovering a signal in compressed sensing. Minimizing k is crucial for these applications, as it corresponds to the communication or number of observations of the underlying algorithm.

A quantity of interest is the operator norm. Given a matrix A, the operator norm $\|A\|_{op}$ is defined to be $\|A\|_{op} = \sup_{x:\|x\|_2=1} \|Ax\|_2$. The operator norm arises in several applications; for example one sometimes approximates a matrix A by another matrix $\hat{A}$ for which $\|A - \hat{A}\|_2$ is small. Often $\hat{A}$ has low rank, in which case this is the low rank approximation problem with spectral error, see, e.g., recent work on this [20]. If one had an estimator for the operator norm of $A - \hat{A}$, one could use it to verify if $\hat{A}$ is a good approximation to A. Given the linearity in the sketching model, if S is sampled from a distribution D, one can compute $S(A) - S(\hat{A}) = S(A - \hat{A})$, from which one then has an estimation procedure to estimate $\|A - \hat{A}\|_2$ as $f(S(A - \hat{A}), S)$. In the sketching model, it was first shown that approximating the operator norm up to a constant factor requires $k = \Omega(d^{3/2})$ [14], which was later improved by Regev to the tight $k = \Omega(d^2)$ [29, Section 6.2]. Note that these lower bounds rule out *any* possible function f as the estimation procedure. It is also implicit in [29, Section 6.2] that approximating the operator norm up to a factor α, where $\alpha - 1 = \Omega(1)$, requires $k = \Omega(d^2/\alpha^6)$. Andoni and Nguyen showed an upper bound of $k = O(d^2/\alpha^4)$ [2], that is, they constructed a distribution $\mathcal{D}$ and corresponding estimation procedure f for which it suffices to set $k = O(d^2/\alpha^4)$. This follows by Theorem 1.2 of [2].

A wide class of matrix norms is the Schatten p-norms, which are the analogues of ℓ_p-norms of vectors and contain the operator norm as a special case. The Schatten p-norm of matrix A is denoted by $\|A\|_p$ and defined to be $\|A\|_p = (\sum_{i=1}^n (\sigma_i(A))^p)^{1/p}$, where $\sigma_1, \ldots, \sigma_n$ are the singular values of A. When $p < 1$, $\|A\|_p$ is not a norm but still a well-defined quantity. For $p = 0$, viewing $\|A\|_0$ as the limit $\lim_{p \to 0^+} \|A\|_p$ recovers exactly the *rank* of A, which has been studied in the data stream [4, 6] and property testing models [11, 16]. When $p = 1$, it is the nuclear or trace norm[1], which has applications in differential privacy [9, 13] and non-convex optimization [5, 8]. When $p = 2$, it is the Frobenius norm, and when $p \to \infty$, it holds that $\|A\|_p$ tends to $\|A\|_{op}$. Such norms are useful in geometry and linear algebra, see, e.g., [29]. A $k = \Omega(\sqrt{d})$ lower bound for every $p \geq 0$ was shown in [15]. For $p > 2$ a lower bound on the sketching dimension of $k = \Omega(d^{2/3-3/p})$, and an upper bound of $k = O(d^{2-4/p})$ were shown in [15]. The upper bound is only known to hold when p is an even integer. The lower bound was improved by Regev to $k = \Omega(d^{2-6/p})$ for $p > 6$ [29, Section 6.2][2].

Other related work includes that on *oblivious subspace embeddings* (OSEs), which fall into the category of bilinear sketches. Here one seeks a distribution $\mathcal{D}$ over $\mathbb{R}^{k\times n}$ such that for any $n \times d$ matrix A,

$$\Pr_{S\sim\mathcal{D}}[\forall x,\ (1-\epsilon)\|Ax\|_2 \leq \|SAx\|_2 \leq (1+\epsilon)\|Ax\|_2] \geq 2/3.$$

This notion has proved important in numerical linear algebra, and has led to the fastest known algorithms for low rank approximation and regression [7, 19, 21]. Since an OSE has the property that $\|SAx\|_2 = (1 \pm \epsilon)\|Ax\|_2$ for all x, it holds in particular that $\|SA\|_{op} = (1 \pm \epsilon)\|A\|_{op}$, where the notation $a = (1 \pm \epsilon)b$ means $(1-\epsilon)b \leq a \leq (1+\epsilon)b$. When $n \geq d/\epsilon^2$, Nelson and Nguyen show the tight bound that any OSE requires $k = \Omega(d/\epsilon^2)$ [22].

Finally, we mention recent related work in the data stream model on approximation of matrix norms [4, 18]. Here one sees elements of A one at a time and the goal is to output an approximation to $\|A\|_p$. It is important to note that the data stream model and sketching models are incomparable. The main reason for this is that unlike in the data stream model, the bit complexity is not accounted for in the sketching model, and both S and A are assumed to have entries which are real numbers. The latter is the common model adopted in compressed sensing. In the data stream model, if one wants to output a vector $v \in \{0, 1, \ldots, M-1, M\}^n$, one needs $n \log M$ bits of space. On the other hand, if u is the vector $(1, (M+1), (M+1)^2, (M+1)^3, \ldots, (M+1)^n)$, then from $\langle u, v\rangle$, one can output v, so the sketching dimension k is only equal to 1. The sketching complexity thus gives a meaningful measure of complexity in the real RAM model. Conversely, lower bounds in the sketching model do not translate into lower bounds in the data stream model. This statement holds even given the work of [14] which characterizes turnstile streaming algorithms as linear sketches. The problem is that lower bounds in the sketching model involve continuous distributions and after discretizing the distributions it is no longer clear if the lower bounds hold.

1.1 Our Contributions

In this paper we strengthen known sketching lower bounds for the operator norm, Schatten p-norms, and subspace embeddings. Our lower bounds are optimal for any approximation to the operator norm, for subspace embeddings, and for Schatten p-norms for even integers p. We first describe our results for the operator norm, as the results for Schatten p-norms and subspace embeddings follow from them.

We consider the following problem: given $\epsilon \in (0, 1/3)$ and $n \geq d/\epsilon^2$, design a distribution

[1] The trace norm is not to be confused with the trace. These two quantities only coincide if A is positive semidefinite.

[2] The section discusses only the case of $p = \infty$, i.e., the operator norm, but the same method can be used for general p and gives the bound claimed here.

$\mathcal{D}$ over $\mathbb{R}^{k\times nd}$ and a function $f:\mathbb{R}^k\times\mathbb{R}^{k\times nd}\to\mathbb{R}$, so that for any $n\times d$ matrix A,

$$\Pr_{S\sim\mathcal{D}}[(1-\epsilon)\|A\|_{op}\le f(S(A),S)\le(1+\epsilon)\|A\|_{op}]\ge 2/3,$$

For this problem, we show a tight $k=\Omega(d^2/\epsilon^2)$ lower bound. Our result considerably strengthens the result of Nelson and Nguyen [22] as it (1) applies only to estimating the operator norm, which can be estimated given any OSE, and (2) applies to general linear sketches rather than only to bilinear sketches. Regarding (1), this shows that designing a general linear sketch for approximating the operator norm of a matrix is *as hard as designing an oblivious subspace embedding.* Regarding (2), we lower bound a much larger class of data structures than OSEs that one could use to approximate $\|Ax\|_2$ for all vectors x.

We then generalize the argument above to handle approximation factors α, with $\alpha-1=\Omega(1)$, for approximating the operator norm. In this case we consider $n=d$, which is without loss of generality since by first applying an OSE S to A with $k=O(d)$, replacing A with $S\cdot A$, all singular values of A are preserved up to a constant factor (we can also pad SA with zero columns to make SA be a square matrix) - see Appendix C of [15]. We can then apply our general linear sketch to SA (the composition of linear sketches is a general linear sketch). We show a lower bound of $k=\Omega(n^2/\alpha^4)$, improving the previous $k=\Omega(n^2/\alpha^6)$ bound, and maching the $k=O(n^2/\alpha^4)$ upper bound. This answers Open Question 2 in [15].

The proof shows the problem is already hard to distinguish between the two cases: (1) A has one singular value of value $\Theta(\alpha)$ and remaining singular values of value $\Theta(1)$, versus (2) all singular values of A are of value $\Theta(1)$. By setting $\alpha=n^{1/p}$, we are able to obtain a constant factor gap in the Schatten-p norm in the two cases, and therefore additionally obtain an $\Omega(n^{2-4/p})$ lower bound for Schatten p-norms for constant factor approximation. This improves the previous $\Omega(n^{2-6/p})$ lower bound, and matches the known upper bound for even integers p. Our proof also establishes a lower bound of $k=\Omega(n^2/s^2)$ for estimating the Ky-Fan s-norm of an $n\times n$ matrix A up to a constant factor, whenever $s\le .0789\sqrt{n}$.

Our main technical novelty is avoiding a deep theorem of Latała [12] concerning tail bounds for Gaussian chaoses used in the prior lower bounds for sketching the operator norm and Schatten p-norms. Instead we prove a simple lemma (Lemma 3) allowing us to bound $\mathbb{E}_{x,y}[e^{x^TAy}]$ for Gaussian vectors x and y and a matrix A, in terms of the Frobenius norm of A. Surprisingly, this lemma suffices for directly upper-bounding the χ^2-distance between the distributions considered in previous works, and without losing any additional factors. Our technical arguments are thus arguably more elementary and simpler than those given in previous work.

2 Preliminaries

Notation

Let $\mathbb{R}^{n\times d}$ be the set of $n\times d$ real matrices and $N(\mu,\Sigma)$ denote the (multi-variate) normal distribution of mean μ and covariance matrix Σ. We write $X\sim\mathcal{D}$ for a random variable X subject to a probability distribution $\mathcal{D}$. Denote by $\mathcal{G}(n,n)$ the ensemble of random matrices with entries i.i.d. $N(0,1)$.

Singular values and matrix norms

Consider a matrix $A\in\mathbb{R}^{n\times n}$. Then A^TA is a positive semi-definite matrix. The eigenvalues of $\sqrt{A^TA}$ are called the singular values of A, denoted by $\sigma_1(A)\ge\sigma_2(A)\ge\cdots\ge\sigma_n(A)$ in decreasing order. Let $r=\operatorname{rank}(A)$. It is clear that $\sigma_{r+1}(A)=\cdots=\sigma_n(A)=0$. Define

$\|A\|_p = (\sum_{i=1}^r (\sigma_i(A))^p)^{1/p}$ $(p > 0)$. For $p \geq 1$, it is a norm over $\mathbb{R}^{n\times d}$, called the p-th *Schatten norm*, over $\mathbb{R}^{n\times n}$ for $p \geq 1$. When $p = 1$, it is also called the trace norm or nuclear norm. When $p = 2$, it is exactly the Frobenius norm $\|A\|_F$. Let $\|A\|_{op}$ denote the operator norm of A when treating A as a linear operator from ℓ_2^n to ℓ_2^n. It holds that $\lim_{p\to\infty} \|A\|_p = \sigma_1(A) = \|A\|_{op}$.

The Ky-Fan s-norm of A, denoted by $\|A\|_{F_s}$, is defined as the sum of the largest s singular values: $\|A\|_{F_s} = \sum_{i=1}^s \sigma_i(A)$. Note that $\|A\|_{F_1} = \|A\|_{op}$ and $\|A\|_{F_s} = \|A\|_1$ for $s \geq r$.

Distance between probability measures

Suppose μ and ν are two probability measures over some Borel algebra $\mathcal{B}$ on $\mathbb{R}^n$ such that μ is absolutely continuous with respect to ν. For a convex function $\phi : \mathbb{R} \to \mathbb{R}$ such that $\phi(1) = 0$, we define the ϕ-divergence

$$D_\phi(\mu||\nu) = \int \phi\left(\frac{d\mu}{d\nu}\right) d\nu.$$

In general $D_\phi(\mu||\nu)$ is not a distance because it is not symmetric.

The *total variation distance* between μ and ν, denoted by $d_{TV}(\mu,\nu)$, is defined as $D_\phi(\mu||\nu)$ for $\phi(x) = |x-1|$. It can be verified that this is indeed a distance.

The χ^2*-divergence* between μ and ν, denoted by $\chi^2(\mu||\nu)$, is defined as $D_\phi(\mu||\nu)$ for $\phi(x) = (x-1)^2$ or $\phi(x) = x^2 - 1$. It can be verified that these two choices of ϕ give exactly the same value of $D_\phi(\mu||\nu)$.

▶ **Proposition 1** ([26, p90]). $d_{TV}(\mu,\nu) \leq \sqrt{\chi^2(\mu||\nu)}$.

▶ **Proposition 2** ([10, p97]). $\chi^2(N(0,I_n) * \mu||N(0,I_n)) \leq \mathbb{E}\, e^{\langle x,x'\rangle} - 1$, *where* $x, x' \sim \mu$ *are independent.*

3 Sketching Lower Bound for $p > 2$

We follow the notations in [15] throughout this section, though the presentation here is self-contained. To start, we present the following lemma.

▶ **Lemma 3.**[3] *Suppose that* $x \sim N(0,I_m)$ *and* $y \sim N(0,I_n)$ *are independent and* $A \in \mathbb{R}^{m\times n}$ *satisfies* $\|A\|_F < 1$. *It holds that*

$$\mathop{\mathbb{E}}_{x,y} e^{x^TAy} \leq \frac{1}{\sqrt{1-\|A\|_F^2}}.$$

Proof. First, it is easy to verify that

$$\begin{aligned}\mathop{\mathbb{E}}_{x,y\sim N(0,1)} e^{axy} &= \frac{1}{2\pi}\iint_{\mathbb{R}\times\mathbb{R}} e^{axy-\frac{x^2+y^2}{2}} dxdy \\ &= \frac{1}{2\pi}\int_{\mathbb{R}}\int_{\mathbb{R}} e^{-\frac{1}{2}(x-ay)^2} e^{-\frac{1}{2}(1-a^2)y^2} dxdy \\ &= \frac{1}{\sqrt{2\pi}}\int_{\mathbb{R}} e^{-\frac{1}{2}(1-a^2)y^2} dy \\ &= \frac{1}{\sqrt{1-a^2}}, \qquad a \in [0,1).\end{aligned}$$

[3] A similar result holds for subgaussian vectors x and y with the right-hand side replaced with $\exp(c\|A\|_F^2)$ for some absolute constant $c > 0$, whose proof requires heavier machinery. We only need the elementary variant here by our choice of hard instance.

Without loss of generality, assume that $m \geq n$. Consider the singular value decomposition $A = U\Sigma V^T$ where U and V are orthogonal matrices of dimension m and n respectively and $\Sigma = \operatorname{diag}\{\sigma_1, \dots, \sigma_n\}$ with $\sigma_1, \dots, \sigma_n$ being the non-zero singular values of A. We know that $\sigma_i \in [0, 1)$ for all i by the assumption that $\|A\|_F < 1$. By rotational invariance of the Gaussian distribution, we may assume that $m = n$ and thus

$$\begin{aligned}
\mathop{\mathbb{E}}_{x,y\sim N(0,I_n)} e^{x^T A y} &= \mathop{\mathbb{E}}_{x,y\sim N(0,I_n)} e^{x^T \Sigma y} \\
&= \frac{1}{(2\pi)^n} \iint_{\mathbb{R}^n \times \mathbb{R}^n} \exp\left\{ \sum_{i=1}^n \left(\sigma_i x_i y_i - \frac{x_i^2 + y_i^2}{2} \right) \right\} dxdy \\
&= \prod_{i=1}^n \frac{1}{\sqrt{1 - \sigma_i^2}} \\
&\leq \frac{1}{\sqrt{1 - \sum_{i=1}^n \sigma_i^2}} \\
&= \frac{1}{\sqrt{1 - \|A\|_F^2}}.
\end{aligned}$$

◀

Next we consider the problem of distinguishing two distributions $\mathcal{D}_1 = \mathcal{G}(m, n)$ and $\mathcal{D}_2$ as defined below. Let $u_1, \dots, u_r$ be i.i.d. $N(0, I_m)$ vectors and $v_1, \dots, v_r$ i.i.d. $N(0, I_n)$ vectors and further suppose that $\{u_i\}$ and $\{v_i\}$ are independent. Let $s \in \mathbb{R}^r$ and define the distribution $\mathcal{D}_2$ as $\mathcal{G}(m, n) + \sum_{i=1}^r s_i u^i (v^i)^T$. We take k linear measurements and denote the corresponding rows (measurements) of the sketching matrix by $L^1, \dots, L^k$. Without loss of generality we may assume that $\operatorname{tr}((L^i)^T L^i)) = 1$ and $\operatorname{tr}((L^i)^T L^j)) = 0$ for $i \neq j$, since this corresponds to the rows of the sketching matrix being orthonormal, which we can assume since we can always change the basis of the row space of the sketching matrix in a post-processing step. Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be the corresponding distribution of the linear sketch of dimension k on $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. The main result is the following theorem.

▶ **Theorem 4.** *There exists an absolute constant $c > 0$ such that $d_{TV}(\mathcal{L}_1, \mathcal{L}_2) \leq 1/10$ whenever $k \leq c/\|s\|_2^4$.*

Proof. It is not difficult to verify that $\mathcal{L}_1 = N(0, I_k)$ and $\mathcal{L}_2 = N(0, I_k) + \mu$, where μ is the distribution of

$$\begin{pmatrix} \sum_{i=1}^r s_i (u^i)^T L^1 v^i \\ \sum_{i=1}^r s_i (u^i)^T L^2 v^i \\ \vdots \\ \sum_{i=1}^r s_i (u^i)^T L^k v^i \end{pmatrix}.$$

Consider a random variable (we shall see in a moment where it comes from)

$$\xi = \sum_{i=1}^k \sum_{j,l=1}^r \sum_{a,c=1}^m \sum_{b,d=1}^n s_j s_l (L^i)_{ab} (L^i)_{cd} (u^j)_a (v^j)_b (u^l)_c (v^l)_d.$$

Take expectation on both sides and notice that the non-vanishing terms on the right-hand side must have $j = l$, $a = c$ and $b = d$,

$$\mathbb{E}\,\xi = \sum_{i=1}^k \sum_{j=1}^r \sum_{a=1}^m \sum_{b=1}^n s_j^2 (L^i)_{ab}^2 \,\mathbb{E}(u^j)_a^2 \,\mathbb{E}(v^j)_a^2 = k\|s\|_2^2.$$

Define an event $\mathcal{E} = \{\|s\|^2\xi < 1/2\}$ and it follows from our assumption and Markov's inequality that $\Pr(\mathcal{E}) \geq 1 - 2c$. Restrict μ to this event and denote the induced distribution by $\tilde{\mu}$. Let $\tilde{\mathcal{L}}_2 = N(0, I_n) + \tilde{\mu}$.

Then the total variation distance between $\mathcal{L}_1$ and $\mathcal{L}_2$ can be upper bounded as

$$\begin{aligned} d_{TV}(\mathcal{L}_1, \mathcal{L}_2) &\leq d_{TV}(\mathcal{L}_1, \tilde{\mathcal{L}}_2) + d_{TV}(\mathcal{L}_2, \tilde{\mathcal{L}}_2) \\ &\leq \sqrt{\mathop{\mathbb{E}}_{z_1,z_2\sim\tilde{\mu}} e^{\langle z_1,z_2\rangle} - 1} + d_{TV}(\mu, \tilde{\mu}) \\ &\leq \sqrt{\frac{1}{\Pr(\mathcal{E})}(\mathop{\mathbb{E}}_{z_1\sim\tilde{\mu},z_2\sim\mu} e^{\langle z_1,z_2\rangle} - 1) + \frac{1}{\Pr(\mathcal{E})} - 1} \end{aligned}$$

and we shall bound $\mathbb{E}\, e^{\langle z_1,z_2\rangle}$ in the rest of the proof.

$$\begin{aligned} \mathop{\mathbb{E}}_{z_1\sim\tilde{\mu},z_2\sim\mu} e^{\langle z_1,z_2\rangle} &= \mathbb{E}\exp\left\{\sum_{i=1}^{k}\sum_{j,a,b}\sum_{j',a',b'} s_j(L^i)_{ab}(u^j)_a(v^j)_b \cdot s_{j'}(L^i)_{a'b'}(x^{j'})_{a'}(y^{j'})_{b'}\right\} \\ &= \mathop{\mathbb{E}}_{u^1,\ldots,u^r,v^1\ldots,v^r|\tilde{\mu}} \prod_{j'=1}^{r} \mathop{\mathbb{E}}_{\substack{x_{j'}\sim N(0,I_m)\\ y_{j'}\sim N(0,I_n)}} \exp\left\{\sum_{a',b'} Q^{j'}_{a',b'}(x^{j'})_{a'}(y^{j'})_{b'}\right\}, \end{aligned}$$

where

$$Q^{j'}_{a',b'} = s_{j'}\sum_{i=1}^{k}\sum_{j,a,b}(L^i)_{ab}(L^i)_{a'b'} \cdot s_j(u^j)_a(v^j)_b.$$

In order to apply the preceding lemma, we need to verify that $\|Q^{j'}\|_F^2 < 1$. Indeed,

$$\begin{aligned} \|Q^{j'}\|_F^2 &= \sum_{a',b'}(Q^{j'})^2_{a',b'} \\ &= s_{j'}^2 \sum_{a',b'}\sum_{i,i'}\sum_{j,a,b}\sum_{\ell,c,d} s_j(L^i)_{ab}(L^i)_{a'b'}(u^j)_a(v^j)_b \cdot s_\ell(L^{i'})_{cd}(L^{i'})_{a'b'}(u^\ell)_c(v^\ell)_d \\ &= s_{j'}^2 \sum_{a',b'}\sum_{i}(L^i)^2_{a'b'}\sum_{j,a,b}\sum_{\ell,c,d} s_j(L^i)_{ab}(u^j)_a(v^j)_b \cdot s_\ell(L^i)_{cd}(u^\ell)_c(v^\ell)_d \\ &\qquad (i \text{ must equal to } i') \\ &= s_{j'}^2 \sum_{i}\sum_{j,a,b}\sum_{\ell,c,d} s_j(L^i)_{ab}(u^j)_a(v^j)_b \cdot s_\ell(L^i)_{cd}(u^\ell)_c(v^\ell)_d \\ &= s_{j'}^2\xi < 1 \end{aligned}$$

since we have conditioned on $\mathcal{E}$. Now it follows from the preceding lemma that

$$\begin{aligned} \mathop{\mathbb{E}}_{u^1,\ldots,u^r,v^1\ldots,v^r} \prod_{i=1}^{r} \mathop{\mathbb{E}}_{x_{j'},y_{j'}} \exp\left\{\sum_{a',b'} Q^{j'}_{a',b'}(x^{j'})_{a'}(y^{j'})_{b'}\right\} &\leq \mathop{\mathbb{E}}_{u^1,\ldots,u^r,v^1\ldots,v^r} \prod_{j'=1}^{r} \frac{1}{\sqrt{1 - s_{j'}^2\xi}} \\ &\leq \mathop{\mathbb{E}}_{u^1,\ldots,u^r,v^1\ldots,v^r} \frac{1}{\sqrt{1 - \|s\|^2\xi}} \\ &\leq 1 + \|s\|^2\,\mathbb{E}\,\xi \\ &\leq 1 + k\|s\|^4, \end{aligned}$$

where, in the third inequality, we used the fact that $1/\sqrt{1-x} \leq 1+x$ for $x \in [0,1/2]$. Therefore,

$$d_{TV}(\mathcal{L}_1, \mathcal{L}_2) \leq \sqrt{\frac{k\|s\|^4}{1-2c}} + \frac{2c}{1-2c} \leq \sqrt{\frac{c}{1-2c}} + \frac{2c}{1-2c} \leq \frac{1}{10}$$

when $c > 0$ is small enough. ◀

We will apply the preceding theorem to obtain our lower bounds for the applications. To do so, notice that by Yao's minimax principle, we can fix the rows of our sketching matrix, and show that the resulting distributions $\mathcal{L}_1$ and $\mathcal{L}_2$ above have small total variation distance. By standard properties of the variation distance, this implies that no estimation procedure f can be used to distinguish the two distributions with sufficiently large probability, thereby establishing our lower bound.

▶ **Corollary 5** (α-approximation to operator norm). *Let $c > 0$ be an arbitrarily small constant. For $\alpha \geq 1 + c$, any sketching algorithm that estimates $\|X\|_{op}$ for $X \in \mathbb{R}^{n \times n}$ within a factor of α with error probability $\leq 1/6$ requires sketching dimension $\Omega(n^2/\alpha^4)$.*

Proof. Let $m = n$ and take $r = 1$ and $s_1 = C\alpha/\sqrt{n}$ for some constant C large enough in $\mathcal{D}_2$ and apply the preceding theorem. ◀

▶ **Corollary 6** (Schatten norms). *There exists an absolute constant $c > 0$ such that any sketching algorithm that estimates $\|X\|_p^p$ ($p > 2$) for $X \in \mathbb{R}^{n \times n}$ within a factor of $1 + c$ with error probability $\leq 1/6$ requires sketching dimension $\Omega(n^{2(1-2/p)})$.*

Proof. Let $m = n$ and take $r = 1$ and $s_1 = 5/n^{1/2-1/p}$ in $\mathcal{D}_2$. Note that $\|X\|_p^p$ differs by a constant factor with high probability when $X \sim \mathcal{D}_1$ and $X \sim \mathcal{D}_2$ (the same hard distribution as in [15]), apply the preceding theorem. ◀

▶ **Corollary 7.** *Let $\epsilon \in (0, 1/3)$. For any matrix $X \in \mathbb{R}^{(d/\epsilon^2) \times d}$, any sketching algorithm that estimates $\|X\|_{op}$ within a factor of $1 + \epsilon$ with error probability $\leq 1/6$ requires sketching dimension $\Omega(d^2/\epsilon^2)$.*

Proof. Let $m = d/\epsilon^2$ and $n = d$. Take $r = 1$ and $s_1 = 3\sqrt{\epsilon/d}$ and apply Theorem 4. Next we shall justify this choice of parameters, that is,

$$G \qquad \text{and} \qquad G + 3\sqrt{\frac{\epsilon}{d}}uv^T$$

differ in operator norm by a factor of $1 + \epsilon$. It follows from the standard result [27] that

$$\|G\|_{op} \leq \frac{\sqrt{d}}{\epsilon} + 1.1\sqrt{d} = (1 + 1.1\epsilon)\frac{\sqrt{d}}{\epsilon}$$

with high probability. Next we shall show that

$$\left\|G + 4\sqrt{\frac{\epsilon}{d}}uv^T\right\|_{op} \geq (1 + 3\epsilon)\frac{\sqrt{d}}{\epsilon},$$

for which it suffices to show that

$$\frac{\left\|Gv + 4\sqrt{\frac{\epsilon}{d}}uv^Tv\right\|_2^2}{\|v\|_2^2} \geq \left((1 + 3\epsilon)\frac{\sqrt{d}}{\epsilon}\right)^2.$$

Expanding the numerator of the left-hand side, we obtain

$$\begin{aligned}
&\frac{\|Gv\|_2^2}{\|v\|_2^2} + \frac{16\epsilon}{d}\|v\|_2^2\|u\|_2^2 + \left\langle Gv, 4\sqrt{\frac{\epsilon}{d}}u \right\rangle \\
&\geq \left(\frac{\sqrt{d}}{\epsilon} - 1.1\sqrt{d}\right)^2 + \frac{16\epsilon}{d}\cdot 0.9^2\frac{d^2}{\epsilon^2} - 4\sqrt{\frac{\epsilon}{d}}\|Gv\|\|u\| \\
&\geq \left((1-1.1\epsilon)^2 + 12.96\epsilon\right)\frac{d}{\epsilon^2} - 4\sqrt{\frac{\epsilon}{d}}\left(\frac{\sqrt{d}}{\epsilon} + 1.1\sqrt{d}\right)\left(1.1\frac{\sqrt{d}}{\epsilon}\right)(1.1\sqrt{d}) \\
&\geq ((1-1.1\epsilon)^2 + 12.96\epsilon)\frac{d}{\epsilon^2} - O\left(\frac{d}{\epsilon^{3/2}}\right) \\
&\geq (1+3\epsilon)^2\frac{d}{\epsilon^2}
\end{aligned}$$

with high probability. ◀

▶ **Corollary 8** (Ky-fan norm). *There exists an absolute constant $c > 0$ such that any sketching algorithm that estimates $\|X\|_{F_s}$ for $X \in \mathbb{R}^{n\times n}$ and $s \leq 0.0789\sqrt{n}$ within a factor of $1 + c$ with error probability $\leq 1/6$ requires sketching dimension $\Omega(n^2/s^2)$.*

Proof. Take $r = s$ and $s_1 = s_2 = \cdots = s_r = 5/\sqrt{n}$ in $\mathcal{D}_2$ and apply Theorem 4, for which we shall show the KyFan s-norms are different with high probability in the two cases.

When $X \sim \mathcal{D}_1$, we know that $\sigma_1(X) \leq 2.1\sqrt{n}$ with high probability and thus $\|X\|_{F_s} \leq 2.1s\sqrt{n}$ with high probability.

When $X \sim \mathcal{D}_2$, we can write $X = G + \frac{5}{\sqrt{n}}P$, where $P = u_1v_1^T + \cdots + u_sv_s^T$. We claim that with high probability $\|P\|_1 \geq 0.9sn$ and thus $\|X\|_{F_s} \geq \frac{5}{\sqrt{n}}\|P\|_{F_s} - \|G\|_{F_s} \geq 4.5s\sqrt{n} - 2.1s\sqrt{n} \geq 2.4s\sqrt{n}$, evincing a multiplicative gap of $\|X\|_{F_s}$ between the two cases.

Now we prove the claim. With high probability, it holds that $0.99\sqrt{n} \leq \|u_i\| \leq 1.01\sqrt{n}$ for all i and $|\sum_{i\neq j}\langle u_i, u_j\rangle| \leq 1.01s\sqrt{n}$. We shall condition on these events below.

By the min-max theorem for singular values,

$$\sigma_\ell^2(P) = \max_{H:\dim H=\ell}\ \min_{\substack{x\in H\\ \|x\|_2=1}} x^TP^TPx,$$

where

$$\begin{aligned}
x^TP^TPx &= \sum_{i,j} x^Tv_iu_i^Tu_jv_j^Tx \\
&= \sum_i x^Tv_iu_i^Tu_iv_i^Tx + \sum_{i\neq j} x^Tv_i(u_i^Tu_j)v_j^Tx \\
&\geq 0.99^2n\sum_i x^Tv_i^Tv_ix - 1.01\sqrt{n}\cdot 1.01k\sqrt{n}\cdot 1.01\sqrt{n} \\
&= 0.99^2n\sum_i x^Tv_i^Tv_ix - 1.01^3kn^{\frac{3}{2}}
\end{aligned}$$

and thus,

$$\begin{aligned}
\sigma_\ell^2(P) &\geq 0.99^2n \max_{H:\dim H=\ell}\ \min_{\substack{x\in H\\ \|x\|_2=1}} \sum_i x^Tv_i^Tv_ix - 1.01^3kn^{\frac{3}{2}} \\
&= 0.99^2n\sigma_\ell^2(V) - 1.01^3kn^{\frac{3}{2}},
\end{aligned}$$

where V is a $k \times n$ matrix with rows $v_1^T, \ldots, v_k^T$. Therefore

$$\|P\|_1 \geq 0.99\sqrt{n}\|V\|_1 - 1.01^{\frac{3}{2}} s^{\frac{3}{2}} n^{\frac{3}{4}}.$$

Since V is a Gaussian random matrix, the classical results imply that $\|V\|_1 \geq 0.99 s\sqrt{n}$ with high probability [25]. The claim follows from our assumption on s. ◀

4 Conclusion

We have presented a simple, surprisingly powerful new analysis which gives optimal bounds on the sketching dimension for a number of previously studied sketching problems, including approximating the operator norm, Schatten norms, and subspace embeddings. We have also presented the first lower bounds for estimating Ky Fan norms. It would be interesting to see if there are other applications of this method to the theory of linear sketches.

References

1 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

2 Alexandr Andoni and Huy L. Nguyen. Eigenvalues of a matrix in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1729–1737, 2013.

3 Alexandr Andoni, Huy L. Nguyên, Yury Polyanskiy, and Yihong Wu. Tight lower bound for linear sketches of moments. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 25–32, 2013.

4 Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *the Proceedings of ESA*, 2015.

5 Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Commun. ACM*, 55(6):111–119, 2012.

6 Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 205–214, 2009.

7 Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 81–90, 2013.

8 Amit Deshpande, Madhur Tulsiani, and Nisheeth K. Vishnoi. Algorithms and hardness for subspace approximation. In *SODA*, pages 482–496, 2011.

9 Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems 25*, pages 2348–2356. 2012.

10 Yuri Ingster and I. A. Suslina. *Nonparametric Goodness-of-Fit Testing Under Gaussian Models*. Springer, 1st edition, 2002.

11 Robert Krauthgamer and Ori Sasson. Property testing of data dimensionality. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 18–27, 2003.

12 Rafał Latała. Estimates of moments and tails of Gaussian chaoses. *Ann. Probab.*, 34(6):2315–2331, 2006.

13 Chao Li and Gerome Miklau. Measuring the achievable error of query sets under differential privacy. *CoRR*, abs/1202.3399, 2012.

14 Yi Li, Huy L. Nguyen, and David P. Woodruff. On sketching matrix norms and the top singular vector. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1562–1581, 2014.

15 Yi Li, Huy L. Nguyen, 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.

16 Yi Li, Zhengyu Wang, and David P. Woodruff. Improved testing of low rank matrices. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'14, New York, NY, USA – August 24-27, 2014*, pages 691–700, 2014.

17 Yi Li and David P. Woodruff. A tight lower bound for high frequency moment estimation with small error. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques – 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, pages 623–638, 2013.

18 Yi Li and David P. Woodruff. On approximating functions of the singular values in a stream. In *STOC*, 2016.

19 Xiangrui Meng and Michael W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 91–100, 2013.

20 Cameron Musco and Christopher Musco. Stronger approximate singular value decomposition via the block lanczos and power methods. *CoRR*, abs/1504.05477, 2015.

21 Jelani Nelson and Huy L. Nguyen. OSNAP: faster numerical linear algebra algorithms via sparser subspace embeddings. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 117–126, 2013.

22 Jelani Nelson and Huy L. Nguyên. Lower bounds for oblivious subspace embeddings. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 883–894, 2014.

23 Eric Price and David P. Woodruff. Applications of the shannon-hartley theorem to data streams and sparse recovery. In *Proceedings of the 2012 IEEE International Symposium on Information Theory, ISIT 2012, Cambridge, MA, USA, July 1-6, 2012*, pages 2446–2450, 2012.

24 Oded Regev. Personal communication, 2014.

25 Terence Tao. *Topics in Random Matrix Theory*. Graduate studies in mathematics. American Mathematical Society, 2012.

26 Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 1st edition, 2008.

27 Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Yonina C. Eldar and Gitta Kutyniok, editors, *Compressed Sensing*, pages 210–268. Cambridge University Press, 2012. Cambridge Books Online. `doi:10.1017/CBO9780511794308.006`.

28 Karl Wimmer, Yi Wu, and Peng Zhang. Optimal query complexity for estimating the trace of a matrix. *CoRR*, abs/1405.7112, 2014.

29 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014.

Bounds on the Norms of Uniform Low Degree Graph Matrices

Dhruv Medarametla[1] **and Aaron Potechin**[2]

1 **Stanford University, Stanford, CA, USA**
`dhruvm321@gmail.com`

2 **Cornell University, Ithaca, NY, USA**
`aaronpotechin@gmail.com`

Abstract

The Sum Of Squares hierarchy is one of the most powerful tools we know of for solving combinatorial optimization problems. However, its performance is only partially understood. Improving our understanding of the sum of squares hierarchy is a major open problem in computational complexity theory.

A key component of analyzing the sum of squares hierarchy is understanding the behavior of certain matrices whose entries are random but not independent. For these matrices, there is a random input graph and each entry of the matrix is a low degree function of the edges of this input graph. Moreoever, these matrices are generally invariant (as a function of the input graph) when we permute the vertices of the input graph. In this paper, we bound the norms of all such matrices up to a polylogarithmic factor.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases sum of squares hierarchy, matrix norm bounds

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.40

1 Introduction

1.1 Background and Motivation

The sum of squares hierarchy, independently developed by Shor, Nesterov, Parrillo, and Lasserre [26, 22, 23, 19], is a powerful tool for solving combinatorial optimization problems. The first level of the sum of squares hierarchy corresponds to semidefinite programming on the input variables, which is extremely useful on its own, and each subsequent level of the sum of squares hierarchy gives a larger but more accurate semidefinite program for the problem.

However, the performance of the sum of squares hierarchy is only partially understood. It is known that the sum of squares hierarchy is strictly more powerful than the Lovasz-Schrijver Hierarchy and the Sherali-Adams hierarchy. It is also known that the sum of squares hierarchy captures the best known algorithms for many problems. For example, the sum of squares hierarchy captures the Goemans-Williamson algorithm for max-cut [11] and the Goemans-Linial relaxation for sparsest cut (which was shown to give an $O(\sqrt{\log n})$ approximation by Arora, Rao, and Vazirani [3]). Also, as shown by Barak, Raghavendra, and Steurer [5] and by Guruswami and Sinop[14], the sum of squares hierarchy captures the sub-exponential algorithm for unique games found by Barak et. al. [2]. That said, for all we know, the sum of squares hierrarchy may do even better than these algorithms on max-cut, sparsest cut, and/or unique games; determining the exact performance of the sum of squares hierrarchy on max-cut, sparsest cut, and unique games is a major open problem.

On the lower bound side, it is known that the sum of squares hierarchy cannot solve NP-hard problems. Such lower bounds generally follow from the result of Grigoriev [12, 13], which was independently rediscovered by Schoenebeck [25], that the sum of squares hierarchy cannot distinguish between a random 3-XOR instance and a random 3-XOR instance with a planted solution. This problem can be reduced to 3-SAT and other NP-hard problems, which implies sum of squares lower bounds for these problems. However, until recently, few lower bounds were known for the sum of squares hierarchy for problems which are not NP-hard. For more information about the sum of squares hierarchy, see the survey of Barak and Steurer [6].

Recently, there have been several papers proving lower bounds for the performance of the Sum Of Squares Hierarchy on the planted clique problem [20, 15, 7, 24]. In the planted clique problem, introduced by Jerrum [16] and Kucera [18], we are given a graph which was created by first choosing a random graph and then randomly planting a clique of size k by choosing k vertices and making them all adjacent to each other. The goal of the problem is to recover the planted clique. Although with high probability the size of the largest clique in a random graph is only around $2 \lg n$, the current best polynomial time algorithm, a spectral algorithm due to Alon et. al. [1], can only solve the planted clique problem for $k = \Theta(\sqrt{n})$. In fact, we have strong reason to believe that doing better than $\Theta(\sqrt{n})$ in polynomial time is hard. It has been shown [16, 8, 9] that several classes of algorithms, including Monte-Carlo Markov chains, the Lovasz-Schrijver Hierarchy, and statistical algorithms, cannot do better than $\Theta(\sqrt{n})$ in polynomial time.

The papers [20, 15, 7, 24] show partial lower bounds on the sum of squares hierarchy for the planted clique problem, proving that the second level of the sum of squares hierarchy cannot solve planted clique if k is much smaller than $\sqrt{n}$ and that the rth level of the sum of squares hierarchy cannot solve planted clique if k is much smaller than $n^{\frac{1}{r+1}}$. While these papers use many different techniques, a crucial part of all of them is probabilistically bounding the norms of certain matrices. In these matrices, the entries are not completely independent of each other, but are low degree in the edges of the input graph and are highly symmetric, so we call them uniform low degree graph matrices.

Here, inspired by these papers [20, 15, 7, 24], we investigate the norms of uniform low degree graph matrices. While special cases of these matrices have been analyzed, here we generalize this analysis, proving bounds on the norms of all uniform low degree graph matrices.

Concurrently with this work, a nearly tight lower bound was proved for the sum of squares hierarchy on the planted clique problem [4], showing that the sum of squares hierarchy cannot solve the planted clique problem in polynomial time if k is much smaller than $\sqrt{n}$. Coming full circle, it turns out that this general analysis of uniform low degree graph matrices is a key component of proving the full lower bound. We have good reason to believe that this analysis of uniform low degree graph matrices will be useful in analyzing the sum of squares hierarchy on other problems and it may also be of independent interest.

Finally, we note that this work can be viewed as progress towards matrix concentration inequalities. In random matrix theory, finding concentration inequalities for the norms of matrix-valued functions is a longstanding open problem. This work gives bounds for the case when the matrix function is highly symmetric and has a random graph as input.

1.2 Preliminaries

In this paper, we use the following standard linear algrebra definitions.

▶ **Definition 1.**
1. Given a matrix M, let $M(i,j)$ be the element in the ith row and jth column of M. We use $M(i,j)$ rather than M_{ij} because we will often want to give our matrices subscripts and superscripts.
2. Given a matrix M, we take $||M||$ to be the induced norm of M, i.e. $||M|| = \max\limits_{||v||=1} ||Mv||$.

Throughout this paper, we will be bounding the norms of matrices whose entries depend on a random graph $G \sim G(n, \frac{1}{2})$. To avoid writing G repeatedly, we make this dependence implicit rather than writing it explicitly.

To bound the norms of our matrices, we will use the moment method. In particular, we use the following fact.

▶ **Lemma 2.** *For any real matrix M, for all $k \geq 1$, $\sqrt[2k]{\mathrm{tr}((MM^T)^k)} \geq ||M||$.*

For completeness, we give a short proof of this fact in Appendix A.

Finally, we recall König's Theorem and Menger's Theorem as they will play a crucial role in our analysis.

▶ **Definition 3.** Given a graph G, a vertex cover of G is a set of vertices $V \subseteq V(G)$ such that all edges of G are incident with at least one vertex in V.

▶ **Theorem 4** (König's Theorem). *If G is a bipartite graph with partite sets U and V then the minimal size of a vertex cover of G is equal to the maximal size of a matching between U and V.*

▶ **Definition 5.** If G is a graph and $U, V \subseteq V(G)$, we define a vertex separator S of U and V to be a set of vertices such that all paths from U to V intersect S.

▶ **Theorem 6** (Menger's Theorem). *If G is a graph and $U, V \subseteq V(G)$ then the minimal size of a vertex separator of U and V is equal to the maximal number of vertex disjoint paths between U and V.*

1.3 Definitions for Uniform Low Degree Graph Matrices

We now rigorously define what uniform low degree graph matrices are. For the remainder of the paper, we assume that $V(G) = [1, n]$ so that the vertices of G have a natural ordering.

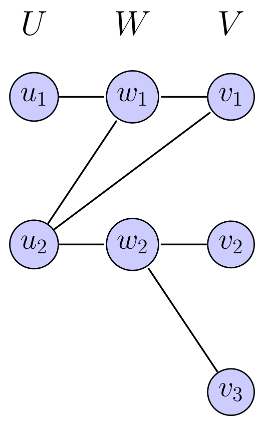
▶ **Definition 7.** Given an input graph G and a possible edge e, we define the edge variable $e = (i,j)$ to be 1 if $(i,j) \in E(G)$ and -1 otherwise. Given a set of edges E, we define $\chi_E = \prod_{e \in E} e$.

▶ Remark. We can think of the χ_E as Fourier characters on the input graph.

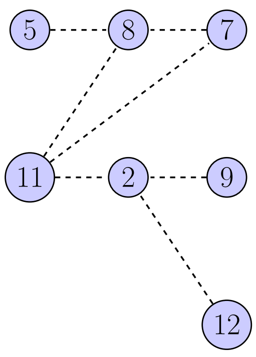
▶ **Definition 8.** We say that a matrix R is a graph matrix if its entries are all functions of the edge variables of some input graph G. We say that R has degree d if the maximum degree among all of these functions is d.

Uniformity says that the matrix is the same (as a function of the input graph G) when we permute the vertices of G. More precisely, we have the following definitions.

▶ **Definition 9.** Given a permutation σ of $V(G)$,
1. If $e = (u,v)$ is a possible edge of G then define $\sigma(e) = (\sigma(u), \sigma(v))$.
2. Given a set E of possible edges of G, define $\sigma(E) = \{\sigma(e) : e \in E\}$.



(a) H.



(b) The subgraph of G that $\chi_{H,A,B,C}$ is calculated from.

Figure 1

▶ **Definition 10.** We say that a graph matrix R is uniform if the following conditions hold:

1. R has rows and columns indexed by subsets A and B of $V(G)$.
2. Letting $c_{A,B,E}$ be the coefficient of χ_E in $R(A,B)$, whenever $A, A', B, B' \subseteq V(G)$, $|A'| = |A|$, $|B'| = |B|$, and σ is a permutation of $V(G)$ which maps the ith element of A to the ith element of A' and maps the jth element of B to the jth element of B', $c_{A',B',\sigma(E)} = c_{A,B,E}$.

In this paper, we focus on the following type of uniform graph matrix.

▶ **Definition 11.** Let H be a graph with two distinguished subsets of vertices $U = \{u_1, u_2, \ldots u_x\}$ and $V = \{v_1, v_2, \ldots v_y\}$. Let $W = \{w_1, \ldots, w_z\}$ be the remaining vertices of H. Given $A = \{a_1, \ldots, a_x\}$, $B = \{b_1, \ldots, b_y\}$, and $C = \{c_1, \ldots, c_z\}$ such that $a_i = b_j$ if and only if $u_i = v_j$, C is disjoint from $A \cup B$, and A and B are in increasing order but C may be in any order (though still with no duplicates), define $\chi_{H,A,B,C} = \chi_{\pi(E(H))}$ where π is the mapping from $V(H)$ to $V(G)$ such that $\forall i \in [1,x], \pi(u_i) = a_i$, $\forall j \in [1,y], \pi(v_j) = b_j$, $\forall k \in [1,z], \pi(w_k) = c_k$ and we take $\pi(E(H)) = \{(\pi(u), \pi(v)) : (u,v) \in E(H)\}$

We define the matrix R_H to be the $\binom{n}{x} \times \binom{n}{y}$ matrix with entries $R_H(A,B) = \sum_C \chi_{H,A,B,C}$ whenever $a_i = b_j$ if and only if $u_i = v_j$ and we take $R_H(A,B) = 0$ otherwise.

▶ **Example 12.** The following is an example of $\chi_{H,A,B,C}$ for a particular H, A, B, and C. If H is the graph shown below in Figure 1a, $A = \{5, 11\}$, $B = \{7, 9, 12\}$, and $C = \{8, 2\}$, then $\chi_{H,A,B,C}$ is calculated from the subgraph of G displayed in Figure 1b. In particular, $\chi_{H,A,B,C}$ is the product of the edge variables of the seven possible edges of G that are displayed in Figure 1b.

▶ Remark. If H is a bipartite graph with partite sets U and V then $R_H(A,B) = 0$ if $A \cap B \neq \emptyset$ and whenever $A \cap B = \emptyset$, $R_H(A,B)$ is ± 1. Moreover, $R(A,B)$ only depends on the edges between A and B in G.

▶ **Example 13.** If H consists of a single edge from u_1 to v_1 then R_H is a ± 1 symmetric random matrix with zeros on the diagonal.

▶ Remark. All uniform graph matrices can be expressed as a linear combination of matrices of the form R_H. Thus, to upper bound the norms of all uniform low degree graph matrices, it is sufficient to upper bound norms of the matrices R_H for small H. To lower bound the norms of all uniform low degree graph matrices, a priori it is insufficient to lower bound the norms of the matrices R_H for small H, as if we take a linear combination of different R_H it is possible that there is almost perfect cancellation between them. That said, it turns out that the probility of such a cancellation is negligible, so the norms of all uniform low degree graph matrices can be understood in terms of the norms of their component R_H. For details on how this can be shown, see Section 6.

1.4 Paper Outline and Results

Our main result is the following theorem.

▶ **Theorem 14.** *Let H be a graph with distinguished sets of vertices U and V such that U and V are disjoint and all vertices in $H(V) \setminus (U \cup V)$ have degree at least one. Let $t = |V(H)|$, let $z = |V(H) \setminus (U \cup V)|$, and let q be the size of the minimal separator between U and V. If $q \geq 1$ then for all $\epsilon \in (0,1)$,*

$$\mathbb{P}\left[||R_H|| \geq 2(t^t)\left(e(t+z)\left(\frac{\ln(8n^q/\epsilon)}{2(q+z)}+1\right)\right)^{q+z} n^{\frac{t-q}{2}}\right] \leq \epsilon\,.$$

In Section 2, we introduce our main techniques by applying them to the simple and well-studied case of a symmetric ± 1 random matrix. We then give a brief technical overview of the proof for the general case in Section 3. In Section 4 we prove the result for all bipartite graphs H with partite sets U and V. In Section 5 we generalize our techniques and prove the full result. The case where U and V have non-trivial intersection is considered in Appendix B. Finally, in Section 6 we show that this theorem is tight up to a polylog(n) factor.

1.5 Comparison with Previous Work

This paper can be compared to the recent body of work [20, 15, 7, 24] showing planted clique lower bounds and to previous work in random matrix theory. In the planted clique lower bounds, $||R_H||$ is bounded for several special cases of H, but only the ones that are needed for the sum of squares lower bounds. In this paper, we use many of the same ideas (constraint graphs, looking at cycles, vertex partitioning), but we consider bounding $||R_H||$ as a mathematical problem independent of its applications to the sum of squares hierarchy, obtaining bounds for all possible H and greatly generalizing the previous work.

In terms of random matrix theory, our results are much less precise than classical results such as Wigner's semicircle law [27] and Girko's circular law [10]. While these results give an exact distribution for the eigenvalues of symmetric random matrices and asymmetric random matrices respectively, we only give a norm bound and this norm bound is off by polylogarithmic factor. That said, the matrices we are considering are much more complicated as the entries are no longer independent and may behave in complex ways on the input graph G. To the best of our knowledge, uniform low degree graph matrices have not previously been studied in random matrix theory. Indeed, as noted in the introduction, obtaining general norm bounds when we have a matrix valued function of random inputs rather than a matrix with independent entries is a longstanding open problem in random matrix theory.

2 Warm-up: Bounding the Norm of a ± 1 Random Matrix

As a warmup, we consider the case of a ± 1 symmetric random matrix. This type of matrix and its norm have already been studied extensively, in particular Wigner's semicircle law

[27] says that with high probability, the norm of an $n \times n$ symmetric random ± 1 matrix is $2\sqrt{n}(1 \pm o(1))$. While our upper bound will not be as strong, it will illustrate the general ideas involved.

▶ **Definition 15.** Given a random graph $G \sim G(n, \frac{1}{2})$ with vertices $1, \cdots, n$, let R be the matrix with the following entries:

$$R(i,j) = \begin{cases} 0 & i = j \\ 1 & (i,j) \in E(G) \\ -1 & (i,j) \notin E(G). \end{cases}$$

▶ Remark. As noted in the introduction, $R = R_H$ where H consists of a single edge from u_1 to v_1.

Note that R is closely related to the adjacency matrix of G; in fact, it is the additive inverse of the Seidel adjacency matrix. Further note that for all $1 \le i, j \le n$, $\mathbb{E}[R(i,j)] = 0$, as any edge (i,j) has probability $\frac{1}{2}$ of being included in G. We now show the following probabilistic bound on the norm of R. Note that this bound has an extra factor of $\ln(n)$, but this is fine for our purposes as in this paper we are only aiming to get the correct norm bounds to within a factor of polylog(n).

▶ **Theorem 16.** *For all $\epsilon \in (0,1)$,*

$$\mathbb{P}\left[||R|| \ge e\sqrt{n}(\ln(n/\epsilon) + 2)\right] \le \epsilon\,.$$

Proof. In order to find a probabilistic bound for $||R||$, we bound $\mathbb{E}\left[\sqrt[2k]{\operatorname{tr}((RR^T)^k)}\right]$. Notice that

$$\operatorname{tr}((RR^T)^k) = \operatorname{tr}(R^{2k}) = \sum_{i_1, i_2, \ldots, i_{2k} \in [1,n]} \left(\prod_{j=1}^{2k} R(i_j, i_{j+1})\right)$$

where $i_{2k+1} = i_1$ and $[1,n] = \{1, 2, \ldots, n\}$. Therefore,

$$\mathbb{E}[\operatorname{tr}((RR^T)^k)] = \mathbb{E}[\operatorname{tr}(R^{2k})] = \mathbb{E}\left[\sum_{i_1, i_2, \ldots, i_{2k} \in [1,n]} \left(\prod_{j=1}^{2k} R(i_j, i_{j+1})\right)\right]$$
$$= \sum_{i_1, i_2, \ldots, i_{2k} \in [1,n]} \mathbb{E}\left[\prod_{j=1}^{2k} R(i_j, i_{j+1})\right]$$

by linearity of expectation. Now, note that because $\mathbb{E}[R(i,j)] = 0$, the vast majority of the terms $\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big]$ are 0; in fact, the only time the expected value is non-zero is when each consecutive pair of i's is distinct and when each $R(i,j)$ term appears an even number of times, in which case the expected value will be 1. Therefore, we can calculate the number of choices for $i_1, i_2, \ldots, i_{2k}$ that yield a non-zero value for $\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big]$ and use that number to bound $\mathbb{E}[\operatorname{tr}((RR^T)^k)]$. We can think of the sum $\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big]$ graphically as a sum over length $2k$ cycles in the vertex set $[1,n]$ where some vertices in the cycle may be equal to each other. We use what we call a **constraint graph** to represent each such cycle (similar graphs appeared in [20] and [15]). In this case, the constraint graph consists of $2k$

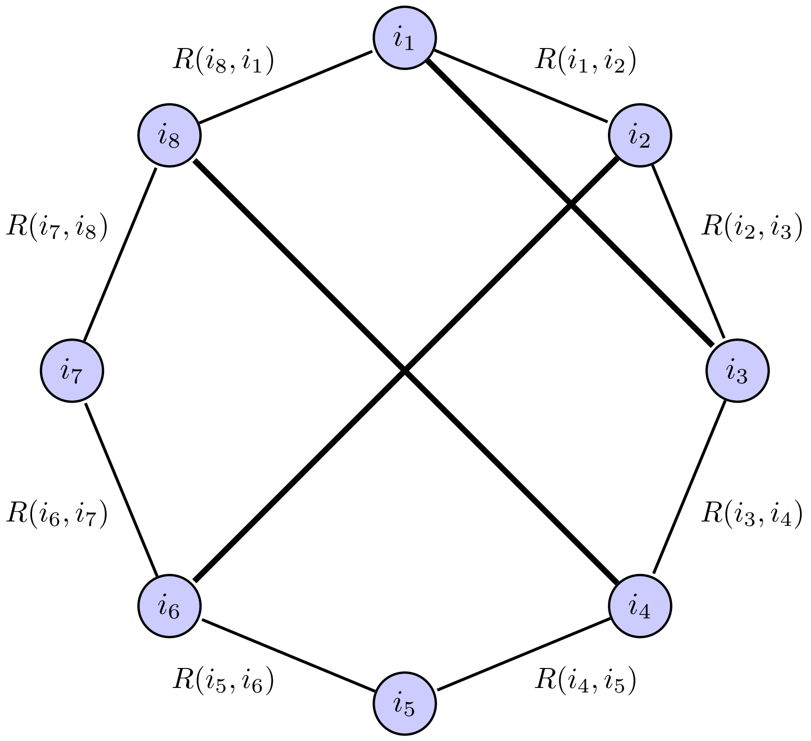


Figure 2 An example of a constraint graph where $k = 4$, $i_1 = i_3$, $i_2 = i_6$, and $i_4 = i_8$.

vertices, each labeled from i_1 to i_{2k}; vertex i_j is connected to vertex i_{j+1} for all $1 \leq j \leq 2k$ to represent the term $R(i_j, i_{j+1})$, and a bold constraint edge is drawn between i_r and i_s whenever $i_r = i_s$ to signify that they are equal.

In the case where j of $2k$ variables are equal, we only draw $j - 1$ constraint edges to represent that equality, rather than $\binom{j}{2}$. This is because each constraint edge essentially represents a restriction; the extra constraint edges do not add to these restrictions, so they are not included.

▶ **Proposition 17.** *In order for* $\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big]$ *to have a non-zero value, there must be at least* $k - 1$ *constraint edges in the respective constraint graph; in addition, this bound is sharp.*

Proof. We prove the first statement by induction on k. When $k = 1$, the statement is vacuously true; $\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big] = \mathbb{E}[R(i_1, i_2)^2]$, which has a non-zero value regardless of constraint edges.

Now, assume that the statement is true for $k = r$, and consider $k = r + 1$. Assume $\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big] \neq 0$, and consider the constraint graph. If each vertex is adjacent to at least one constraint edge, then because each constraint edge is incident to two vertices, there are at least $\frac{2r+2}{2} = r + 1$ constraint edges, and we are done. Therefore, we only need to consider the case where there exists a vertex that is not adjacent to any constraint edges. Call this vertex i_j. Then, note that the statement $i_{j-1} = i_{j+1}$ must be true; if it was not, then the values $R(i_{j-1}, i_j)$ and $R(i_j, i_{j+1})$ have no corresponding equal terms, which means

$\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big] = 0$. But if $i_{j-1} = i_{j+1}$, then $R(i_{j-1}, i_j) = R(i_j, i_{j+1})$, meaning that we no longer need to consider the vertex i_j and its adjacent edges. Therefore, we can treat the vertices i_{j-1} and i_{j+1} as the same vertex, as they are equal, meaning that we have essentially reduced the constraint graph to one on $2r$ vertices. Then, by our induction hypothesis, this constraint graph requires at least $r-1$ constraint edges to create a nonzero expected value, which means that our total constraint graph requires at least r constraint edges, completing the proof.

In order to prove the sharpness of the bound, simply consider the case where $i_j = i_{2k+2-j}$ for all $2 \le j \le k$. Then, $R(i_l, i_{l+1}) = R(i_{2k+1-l}, i_{2k+2-l})$ for all $1 \le l \le k$, which creates a non-zero expected value. ◀

We now use Proposition 17 to bound the maximum number of times that $\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big]$ can take a non-zero value, and use that information to bound $\mathbb{E}[\mathrm{tr}(R^{2k})]$.

▶ **Proposition 18.** *Given a constraint graph on b vertices such that at least c constraint edges are required to create a non-zero expectation value, where each vertex has n possible values, let N represent the number of choices for the b vertices such that the expectation value of the product is non-zero. Then, $N \le \binom{b}{c} n^{b-c}(b-c)^c \le b^{2c} n^{b-c}$.*

Proof. Treat the set of vertices as an ordered set $S = \{d_1, d_2, \ldots, d_b\}$.

Because there must be at least c constraint edges, there must be at least c elements of S that are duplicates of other elements, so we can choose a set $I \subseteq S_b$ of c indices such that for all $j \in I$, there exists $m \notin I$ such that $d_j = d_m$. There are $\binom{b}{c}$ choices for I. We can then choose the elements $\{d_j \mid j \notin I\}$. Each element has at most n possible values so there are at most n^{b-c} choices for these elements. Finally, we choose the elements $\{d_j \mid j \in I\}$. To determine each d_j it is enough to specify the $m \notin I$ such that $d_j = d_m$. Each such d_j has $b-c$ choices, so there are at most $(b-c)^c$ choices for these elements. Therefore, $N \le \binom{b}{c} n^{b-c}(b-c)^c$.

Now, note that $\binom{b}{c} \le b^c$, as $\binom{b}{c} = \frac{b!}{(b-c)!c!} \le \frac{b!}{(b-c)!} \le b^c$. As $(b-c)^c \le b^c$, this completes the proof. ◀

▶ **Corollary 19.** *Let N represent the number of choices for the variables $(i_1, i_2, \ldots, i_{2k})$ such that $\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big] \ne 0$. Then, $N \le (2k)^{2k-2} n^{k+1}$.*

Proof. Apply Proposition 18. Note that $b = 2k$ and $c = k-1$ by Proposition 17. This implies the desired result. ◀

▶ **Corollary 20.** $\mathbb{E}[\mathrm{tr}(R^{2k})] \le (2k)^{2k-2} n^{k+1}$.

Proof. Recall $\mathbb{E}[\mathrm{tr}(R^{2k})] = \sum_{i_1, i_2, \ldots, i_{2k} \in [1,n]} \mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big]$. By Corollary 19, the number of choices for $(i_1, i_2, \ldots, i_{2k})$ that yield a non-zero value for $\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big]$ is at most $(2k)^{2k-2} n^{k+1}$; in addition, $\mathbb{E}\big[\prod_{j=1}^{2k} R(i_j, i_{j+1})\big] \le 1$ for all choices of $(i_1, i_2, \ldots, i_{2k})$. These two observations complete the proof. ◀

Now, note that for any matrix R, $\mathrm{tr}(R^{2k})$ must take on a nonnegative value. By Markov's inequality, for all $\epsilon \in (0,1)$ and all $k \geq 1$, $\mathbb{P}[\mathrm{tr}(R^{2k}) \geq \frac{\mathbb{E}[\mathrm{tr}(R^{2k})]}{\epsilon}] \leq \epsilon$

Using Corollary 20, $\mathbb{P}[\mathrm{tr}(R^{2k}) \geq (2k)^{2k-2}n^{k+1}/\epsilon] \leq \epsilon$. Since $||R|| \leq \sqrt[2k]{\mathrm{tr}((RR^T)^k)} = \sqrt[2k]{\mathrm{tr}(R^{2k})}$ for all $k \geq 1$, this implies that for all $\epsilon \in (0,1)$ and all $k \geq 1$,

$$\mathbb{P}\left[||R|| \geq \sqrt[2k]{(2k)^{2k-2}n^{k+1}/\epsilon}\right] \leq \epsilon\,.$$

Choosing $k = \lceil \ln(n/\epsilon)/2 \rceil$, we have that

$$\sqrt[2k]{(2k)^{2k-2}n^{k+1}/\epsilon} \leq \sqrt[2k]{(2k)^{2k}n^{k+1}/\epsilon} = 2k\sqrt{n}(n/\epsilon)^{\frac{1}{2k}} = 2k\sqrt{n}e^{\frac{\ln(n/\epsilon)}{2k}} \leq e\sqrt{n}(\ln(n/\epsilon)+2)\,.$$

Thus, $\mathbb{P}[||R|| \geq e\sqrt{n}(\ln(n/\epsilon)+2)] \leq \epsilon$, as needed. ◀

In the following sections, we generalize these techniques for matrices whose entries depend on the random graph in more complex ways.

3 Technical Overview of the General Norm Bounds

For the general bounds on $||R_H||$, we use similar ideas. The following is almost correct, but there is a technical issue that needs to be dealt with which we discuss afterwards. We express $E[tr((R_H {R_H}^T)^k)]$ as a sum of many different terms, each of which can be represented with a constraint graph. We upper bound the number of terms which have nonzero expectation by showing a lower bound on the number of constraint edges needed. We then use this to probabilistically bound $||R_H||$.

In the case where H is bipartite, each vertex of H has k copies in the constraint graph so the total number of vertices is kt where $t = V(H)$. The number of constraint edges that are needed to make a term have non-zero expectation is $q(k-1)$ where q is the size of a minimal vertex cover of H. One way we can achieve this is as follows. We take a minimal vertex cover S of H and set all copies of a vertex in S to be equal to each other. Since each vertex in H is copied k times, this requires $q(k-1)$ constraint edge. It turns out that this is tight. Using this bound, there are at most $O(n^{tk-q(k-1)})$ nonzero terms in $E[tr((R_H {R_H}^T)^k)]$ (where the constant hides a function of k). Taking this to the power $\frac{1}{2k}$ for an appropriately chosen k, we obtain that with high probability, $||R_H||$ is at most $O(n^{\frac{t-q}{2}}polylog(n))$. The general case is more complicated but similar ideas apply. It turns out that the key object is a minmal separator S of U and V in H.

However, there is a technical issue in the analysis. In order to obtain the lower bounds on the number of constraint edges needed, we need to assume that the constraint edges behave nicely, namely that we don't have constraint edges between copies of two different vertices in H. This makes part of the constraint graph decompose into disjoint cycles, allowing us to use Proposition 17 (without this restriction, we could have constraint edges between different cycles, which invalidates the analysis). To handle this, we use a vertex partitioning argument. In particular, given a partition $V_1, \ldots, V_t$ of $[1,n]$ we consider the part of R_H where for all i, vertex i is in V_i. This gives us a matrix R' where when we look at $E[tr((R'{R'}^T)^k)]$, the constraint edges behave nicely and we can obtain a probabilistic bound on $||R'||$. We then bound $||R_H||$ using the bound on $||R'||$.

4 Bounding the Norms of Uniform Locally Random Matrices

In this section, we generalize the techniques used in Section 2 to prove Theorem 14 whenever H is a bipartite graph with partite sets U and V. We call these matrices locally random because the value of the entry in row A and column B only depends on the behavior of the input graph G on the vertices $A \cup B$.

▶ **Theorem 21.** *If H is a bipartite graph with t vertices and minimal vertex cover of size q then*

1. $||R_H|| \leq n^{\frac{t}{2}}$
2. *If $q \geq 1$, for all $\epsilon \in (0,1)$,*

$$\mathbb{P}\left[||R_H|| > 2t^t \left(et\left(\frac{\ln(8n^q/\epsilon)}{2q} + 1\right)\right)^q n^{\frac{t-q}{2}}\right] < \epsilon$$

▶ Remark. As we will show in Section 6, this bound is tight up to a factor of polylog(n).

Proof. For the first statement, recall that for any matrix M, $||M|| \leq ||M||_{Fr}$, where $||M||_{Fr} = \sqrt{\sum_{i,j} M(i,j)^2}$ is the Frobenius norm of M. To see this, note that if u and v are unit vectors then

$$u^T M v = \sum_{i,j} u_i M(i,j) v_j \leq \sqrt{\sum_{i,j} {u_i}^2 {v_j}^2} \sqrt{\sum_{i,j} M(i,j)^2} = \sqrt{\sum_{i,j} M(i,j)^2}$$

by the Cauchy-Schwarz inequality. Since every entry of R_H has magnitude at most 1, the result follows.

For the second statement, as described in the technical overview, we first bound the norms of closely related matrices where we restrict which vertices H can map into. We will then use this bound to bound $||R_H||$.

▶ **Definition 22.** Given a partition $V_1, \ldots, V_t$ of the vertices of $V(G)$, we define $R_{H,V_1,\ldots,V_t}$ be the $\binom{n}{x} \times \binom{n}{y}$ matrix such that

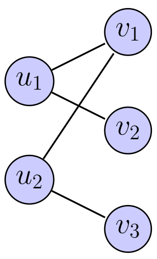
$$R_{H,V_1,\ldots,V_t}(A,B) = \begin{cases} R_H(A,B) = \chi_{H,A,B} & A \cap B = \emptyset, \\ & \forall i \in [1,x], a_i \in V_i,\ \forall j \in [1,y], b_j \in V_{x+j} \\ 0 & \textit{otherwise} \end{cases}$$

▶ **Lemma 23.** *Let $R' = R_{H,V_1,\ldots,V_t}$. For all $\epsilon \in (0,1)$,*

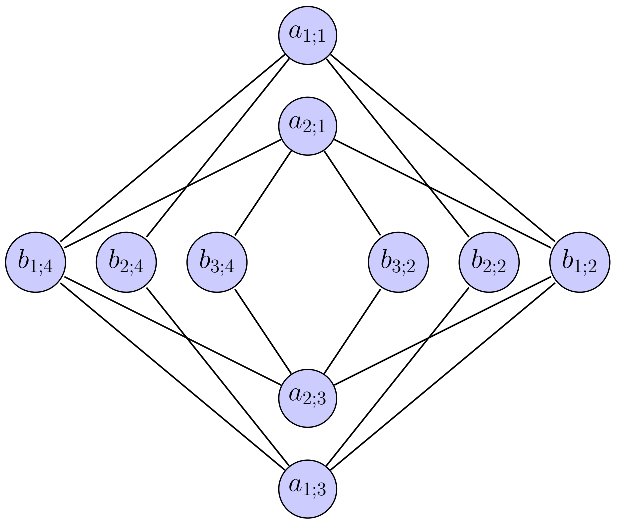
$$\mathbb{P}\left[||R'|| \geq \left(et\left(\frac{\ln(n^q/\epsilon)}{2q} + 1\right)\right)^q n^{\frac{t-q}{2}}\right] \leq \epsilon .$$

Proof. As before, we probabilistically bound $||R'||$ by bounding $\mathbb{E}[\sqrt[2k]{\mathrm{tr}((R'R'^T)^k)}]$. Define $\binom{[n]}{i}$ to be the set of all subsets of $[1,n]$ of size i. Now note that

$$\begin{aligned}\mathbb{E}[\mathrm{tr}((R'R'^T)^k)] &= \mathbb{E}\left[\sum_{\substack{A_1,A_3,\ldots,A_{2k-1} \in \binom{[n]}{x} \\ B_2,B_4,\ldots,B_{2k} \in \binom{[n]}{y}}} \left(\prod_{j=1}^{k} R'(A_{2j-1},B_{2j}) R'^T(B_{2j},A_{2j+1})\right)\right] \\ &= \sum_{\substack{A_1,A_3,\ldots,A_{2k-1} \in \binom{[n]}{x} \\ B_2,B_4,\ldots,B_{2k} \in \binom{[n]}{y}}} \mathbb{E}\left[\prod_{j=1}^{k} R'(A_{2j-1},B_{2j}) R'^T(B_{2j},A_{2j+1})\right]\end{aligned}$$



(a) H.



(b) An example of the constraint graph for the given example of H, where $k = 2$.

Figure 3

by linearity of expectation. Denote $\prod_{j=1}^{k} R'(A_{2j-1}, B_{2j})R'^T(B_{2j}, A_{2j+1})$ as $P(A_1, \ldots, B_{2k})$. Similarly to the previous case, because $\mathbb{E}[R'(A,B)] = 0$, the vast majority of the terms $\mathbb{E}[P(A_1, \ldots, B_{2k})]$ are 0; the only time the expected value can be non-zero is when each consecutive pair of A's and B's is disjoint and every edge of G involved in the product appears an even number of times. In this case, the expected value will be 1. So, we can bound the number of choices for $A_1, B_2, \ldots, A_{2k-1}, B_{2k}$ that yield a non-zero value for $\mathbb{E}[P(A_1, \ldots, B_{2k})]$ and use that number to bound $\mathbb{E}[\text{tr}((R'R'^T)^k)]$. In order to represent $\mathbb{E}[P(A_1, \ldots, B_{2k})]$, we use another constraint graph.

This constraint graph is similar to the constraint graph in Proposition 17. In this constraint graph, there are $k(x + y)$ vertices sorted into $2k$ sets. These vertices are labeled $A_1 = \{a_{1;1}, a_{2;1}, \ldots, a_{x;1}\}, B_2 = \{b_{1;2}, b_{2;2}, \ldots, b_{y;2}\}, A_3 = \{a_{1;3}, a_{2;3}, \ldots, a_{x;3}\}, \ldots, B_{2k} = \{b_{1;2k}, b_{2;2k} \ldots, b_{y;2k}\}$. Two vertices $a_{p;q}$ and $b_{r;s}$ are adjacent in the constraint graph if and only if $|q - s| = 1$ and u_p and v_r are adjacent in H, where $a_{p;1} = a_{p;2k+1}$.

Now, in order to bound the number of choices for $A_1, B_2, \ldots, A_{2k-1}, B_{2k}$ that yield a non-zero expectation value, we can introduce the constraint edges again. However, note that due to the definition of R', constraint edges can only exist between vertices of the constraint graph that are created by the same vertex of H, as it is impossible for two vertices that are not created by the same vertex of H to be equal, as they correspond to different disjoint sets V_i and the value of each variable must be in its respective set.

▶ **Lemma 24.** *In order for $\mathbb{E}[P(A_1, \ldots, B_{2k})]$ to have a non-zero value, there must be at least $q(k - 1)$ constraint edges in the respective constraint graph, where q is the size of a minimal vertex cover of H; in addition, this bound is sharp.*

Proof. In order to prove this lemma, we first show that the given bound is an upper bound then show that it is sharp by König's Theorem [17].

First, note that in order for $\mathbb{E}[P(A_1, \ldots, B_{2k})]$ to have a non-zero value, every edge in the constraint graph must have an equal counterpart by virtue of the constraint edges; this ensures that any edge that appears in the product appears an even number of times, creating a non-zero expected value.

It is easy to see that at most $q(k-1)$ constraint edges are required; namely, if V is a minimal vertex cover of H, then if $x_i \in V$, set $a_{i;1} = a_{i;3} = \cdots = a_{i;2k-1}$, and if $y_j \in V$, set $b_{j;2} = b_{j;4} = \cdots = b_{j;2k}$. Each such set of equalities corresponds to $k-1$ constraint edges, meaning that there are $q(k-1)$ constraint edges total. In addition, every edge in the constraint graph will have an equal counterpart by this method. If $(x_i, y_j) \in H$, at least one of x_i and y_j is in V by definition; without loss of generality $y_j \in V$. Then, this implies that for the edges in the constraint graph of the form $(a_{i;1}, b_{j;2}), (b_{j;2}, a_{i;3}), (a_{i;3}, b_{j;4}), \dots, (b_{j;2k}, a_{i;1})$, each edge $(a_{i;2m-1}, b_{j,2m-2})$ has the equal counterpart $(a_{i;2m-1}, b_{j,2m})$ for $1 \le m \le k$, as $b_{j,2m-2} = b_{j,2m}$. Thus, this set of constraint edges is sufficient to create a non-zero expected value.

Now, we must show at least $q(k-1)$ constraint edges are required. Because H is a bipartite graph, we can apply König's Theorem, which states that there exists a matching of size q in H. Consider the q disjoint cycles of length $2k$ in the constraint graph that are created by the q edges in the matching of H. Because R' is defined so that constraint edges can only exist between vertices $a_{p;q}$ and $a_{p;q'}$ or between $b_{r;s}$ and $b_{r;s'}$, as two vertices not of this form do not belong to the same set V_i, any constraint edge created can affect at most 1 of the q cycles, due to the fact that all the cycles are disjoint and thus are impossible to link with a constraint edge. Therefore, each cycle requires at least $k-1$ constraint edges by Proposition 17, implying that the q cycles require at least $q(k-1)$ constraint edges total, completing the proof. ◀

▶ **Corollary 25.** *Let N represent the number of choices for the sets $A_1, B_2, \dots, A_{2k-1}, B_{2k}$ such that $\mathbb{E}[P(A_1, \dots, B_{2k})] \neq 0$. Then, $N \le (tk)^{2(k-1)q} n^{(t-q)k+q}$ where $t = |V(H)| = x + y$.*

Proof. Apply Proposition 18. In this situation, $b = kt$ and $c = q(k-1)$. This implies the desired result. ◀

▶ **Corollary 26.** $\mathbb{E}\left[\operatorname{tr}((R'R'^T)^k)\right] \le (tk)^{2(k-1)q} n^{(t-q)k+q}$.

Proof. Recall $\mathbb{E}[\operatorname{tr}((R'R'^T)^k)] = \sum_{\substack{A_1, A_3, \dots A_{2k-1} \in \binom{[n]}{x} \\ B_2, B_4, \dots B_{2k} \in \binom{[n]}{y}}} \mathbb{E}\left[\prod_{j=1}^{k} R'(A_{2j-1}, B_{2j}) R'^T(B_{2j}, A_{2j+1})\right]$.
Then, by Proposition 25, the number of choices for $A_1, B_2, \dots A_{2k-1}, B_{2k}$ that yield a non-zero value for $\mathbb{E}\left[\prod_{j=1}^{k} R'(A_{2j-1}, B_{2j}) R'^T(B_{2j}, A_{2j+1})\right]$ is at most $(tk)^{2(k-1)q} n^{(t-q)k+q}$; in addition,
$\mathbb{E}\left[\prod_{j=1}^{k} R'(A_{2j-1}, B_{2j}) R'^T(B_{2j}, A_{2j+1})\right] \le 1$. These two observations complete the proof. ◀

Now, note that for any graph G on n vertices, $\operatorname{tr}((R'R'^T)^k)$ must take on a nonnegative value. Then, by Markov's inequality and Corollary 26, for all $\epsilon \in (0,1)$,

$$\mathbb{P}\left[\operatorname{tr}((R'R'^T)^k) \ge \frac{\mathbb{E}[\operatorname{tr}((R'R'^T)^k)]}{\epsilon}\right] \le \mathbb{P}\left[\operatorname{tr}((R'R'^T)^k) \ge (tk)^{2(k-1)q} n^{(t-q)k+q}/\epsilon\right] \le \epsilon.$$

Since $||R'|| \le \sqrt[2k]{\operatorname{tr}((R'R'^T)^k)}$, this implies that for all $k \ge 1$ and all $\epsilon \in (0,1)$,

$$\mathbb{P}\left[||R'|| \ge \sqrt[2k]{(tk)^{2(k-1)q} n^{(t-q)k+q}/\epsilon}\right] \le \mathbb{P}\left[||R'|| \ge (tk)^q n^{\frac{t-q}{2}} (n^q/\epsilon)^{1/2k}\right] \le \epsilon\,.$$

Setting $k = \lceil \frac{1}{2q}\ln(n^q/\epsilon)\rceil$ we have that $(tk)^q n^{\frac{t-q}{2}}(n^q/\epsilon)^{1/2k} \leq \left(t\left(\frac{\ln(n^q/\epsilon)}{2q}+1\right)\right)^q n^{\frac{t-q}{2}}e^q$ Therefore, $\mathbb{P}[||R'|| \geq \left(et\left(\frac{\ln(n^q/\epsilon)}{2q}+1\right)\right)^q n^{\frac{t-q}{2}}] \leq \epsilon$, as needed. ◀

We can now use our bounds for $||R'||$ to bound $||R||$ through the following lemma.

▶ **Lemma 27.** *Let M be a matrix and B, p be positive numbers such that:*

1. *$M = \frac{1}{N}\sum_{V_1,\cdots,V_t} M_{V_1,\cdots,V_t}$ for some matrices $\{M_{V_1,\cdots,V_t}\}$ where N is the number of possible $V_1,\cdots,V_t$.*
2. *For each choice of $V_1,\cdots,V_t$, for all $x \in [\frac{1}{2}, N]$, $\mathbb{P}(||M_{V_1,\cdots,V_t}|| > Bx) \leq \frac{p}{64x^3}$.*

then $\mathbb{P}(||M|| \geq B) < p$.

▶ Remark. Unlike in [15], we use all possible partitions so we have that $N = t^n$

Proof. The result follows from the following proposition.

▶ **Proposition 28.** *For all $j \in [0, \lg N]$, the probability that there are more than $\frac{N}{2^{2j+2}}$ matrices $M_{V_1,\cdots,V_t}$ such that $||M_{V_1,\cdots,V_t}|| > 2^{j-1}B$ is at most $\frac{p}{2^{j+1}}$.*

Proof. We prove this by contradiction. If the probability that there are more than $\frac{N}{2^{2j+2}}$ matrices $M_{V_1,\cdots,V_t}$ such that $||M_{V_1,\cdots,V_t}|| > 2^{j-1}B$ is greater than $\frac{p}{2^{j+1}}$ then the probability that $||M_{V_1,\cdots,V_t}|| > 2^{j-1}B$ must be greater than $\frac{p}{2^{3j+3}}$. Plugging in $x = 2^{j-1}$, this gives a contradiction. ◀

Using this proposition, with probability at least $1 - \sum_{j=0}^{\lfloor \lg n \rfloor} \frac{p}{2^{j+1}}$, for all integers j such that $0 \leq j \leq \lg N$, there are at most $\frac{N}{2^{2j+2}}$ matrices $M_{V_1,\cdots,V_t}$ such that $||M_{V_1,\cdots,V_t}|| > 2^{j-1}B$. When this occurs, for all integers j such that $0 \leq j \leq \lfloor \lg N \rfloor - 1$, there are at most $\frac{N}{2^{2j+2}}$ matrices $M_{V_1,\cdots,V_m}$ such that $2^{j-1}B < ||M_{V_1,\cdots,V_m}|| \leq 2^j B$. Moreover, there are no matrices such that $||M_{V_1,\cdots,V_t}|| > 2^{\lfloor \lg N \rfloor - 1}B$. This implies that with probability at least $1 - \sum_{j=0}^{\lfloor \lg n \rfloor} \frac{p}{2^{j+1}}$, $||M|| \leq \frac{B}{2} + \sum_{j=0}^{\lfloor \lg N \rfloor} \frac{2^j B}{2^{2j+2}} < B$, as needed. Since $1 - \sum_{j=0}^{\lfloor \lg n \rfloor} \frac{p}{2^{j+1}} > 1 - p$, the result follows. ◀

▶ **Proposition 29.** *Set $M = R_H$ and $M_{V_1,\ldots,V_t} = t^t R_{H,V_1,\ldots,V_t}$. For all $\epsilon \in (0,1)$, take $p = \epsilon$ and let $B = 2t^t\left(et\left(\frac{\ln(8n^q/\epsilon)}{2q}+1\right)\right)^q n^{\frac{t-q}{2}}$. Then, conditions (1) and (2) of Lemma 27 holds true for these values of M, $M_{V_1,\ldots,V_t}$, B, and p.*

Proof. We must show both (1) and (2).

Note that $M = \frac{1}{N}\sum_{V_1,\ldots,V_t} M_{V_1,\ldots,V_t}$ because given a non-zero term $R_H(A,B)$, $R'(A,B)$ has probability $\frac{1}{t^t}$ of equaling $R_H(A,B)$ among all possible $V_1, V_2, \ldots, V_t$. Thus, part (1) of the Lemma holds.

Now, we must show (2); that $\mathbb{P}[t^t||R'|| > Bx] \leq \frac{p}{64x^3}$ for all $x \in [\frac{1}{2}, N]$. Plugging in $\epsilon' = \frac{\epsilon}{64x^3}$ to Lemma 23, we have that

$$\mathbb{P}\left[||R'|| \geq \left(et\left(\frac{\ln(64x^3n^q/\epsilon)}{2q}+1\right)\right)^q n^{\frac{t-q}{2}}\right] \leq \frac{\epsilon}{64x^3}.$$

We need to show that for all $x \in [\frac{1}{2}, N]$, $Bx \geq t^t\left(et\left(\frac{\ln(64x^3n^q/\epsilon)}{2q}+1\right)\right)^q n^{\frac{t-q}{2}}$. Note that if $x = \frac{1}{2}$ then $Bx = t^t\left(et\left(\frac{\ln(64x^3n^q/\epsilon)}{2q}+1\right)\right)^q n^{\frac{t-q}{2}}$ so it is sufficient to show that

$\frac{\left(\frac{\ln(64x^3n^q/\epsilon)}{2q}+1\right)^q}{x}$ is a decreasing function for $x \geq \frac{1}{2}$. Taking the derivative of this function yields

$$\frac{\left(\frac{\ln(64x^3n^q/\epsilon)}{2q}+1\right)^q}{x^2}\left(\frac{3}{2\left(\frac{\ln(64x^3n^q/\epsilon)}{2q}+1\right)}-1\right)$$

This is negative for $x \geq \frac{1}{2}$ if $n \geq e$. If $n \leq e$ then it only makes sense to have $q \leq 1$ and we again have that this is negative for $x \geq \frac{1}{2}$. This completes the proof. ◀

Now that we know our particular values of M, $M_{V_1,\ldots,V_t}$, B, and p satisfy the conditions of Lemma 27, we apply the aforementioned lemma, obtaining that

$$\mathbb{P}\left[||R_H|| > 2t^t\left(et\left(\frac{\ln(8n^q/\epsilon)}{2q}+1\right)\right)^q n^{\frac{t-q}{2}}\right] < \epsilon\,.$$

as needed. ◀

5 Bounding the Norms of Uniform Low Degree Graph Matrices

In this section, we generalize our techniques further to prove our main result, Theorem 14, which we restate here.

▶ **Theorem 30.** *Let H be a graph with distinguished sets of vertices U and V such that U and V are disjoint and all vertices in $H(V)\setminus(U\cup V)$ have degree at least one. Let $t = |V(H)|$, let $z = |V(H)\setminus(U\cup V)|$, and let q be the size of the minimal separator between U and V.*

1. *If $q+z \geq 1$ then for all $\epsilon \in (0,1)$,*

$$\mathbb{P}\left[||R_H|| \geq 2(t^t)\left(e(t+z)\left(\frac{\ln(8n^q/\epsilon)}{2(q+z)}+1\right)\right)^{q+z} n^{\frac{t-q}{2}}\right] \leq \epsilon\,.$$

2. *If $q = z = 0$ then $||R_H|| \leq n^{\frac{t}{2}}$.*

Proof. For the second statement, note that if $q = z = 0$ then every entry of R_H has magnitude at most 1, so we can again use the fact that $||R_H|| \leq ||R_H||_{Fr} = \sqrt{\sum_{A,B} R_H(A,B)^2}$. For the first statment, similar to before, we first bound the norm of a closely related matrix where we restrict which entries the vertices of H can map into.

▶ **Definition 31.** Let $W = w_1,\ldots,w_z$ be the vertices of H outside of U and V. Given a partition $V_1,\ldots,V_t$ of $V(G)$, define $R_{H,V_1,\ldots,V_t}$ to be the $\binom{n}{x}\times\binom{n}{y}$ matrix with entries

$$R_{H,V_1,\ldots,V_t}(A,B) = \begin{cases} \sum_{C:\forall k, c_k \in V_{x+y+k}} \chi_{H,A,B,C} & A\cap B = \emptyset, \\ & \forall i\in[1,x], a_i \in V_i, \forall j \in [1,y], b_j \in V_{x+j} \\ 0 & \textit{otherwise} \end{cases}$$

where the sum is over all $C = \{c_1,\ldots,c_z\}$ where the $c_1,\ldots,c_z$ are disjoint but not necessarily in order.

Let $R' = R_{H,V_1,\ldots,V_t}$. In order to find a probabilistic bound for $||R'||$, we bound $\mathbb{E}[\sqrt[2k]{\mathrm{tr}((R'R'^T)^k)}]$.

▶ **Lemma 32.** $\mathbb{P}\left[||R'|| \geq \left(e(t+z)\left(\frac{\ln(n^q/\epsilon)}{2(q+z)}+1\right)\right)^{q+z} n^{\frac{t-q}{2}}\right] \leq \epsilon.$

Proof.

▶ **Definition 33.** Define $S_{n,z}$ to be the set of all ordered tuples of z distinct elements of $[1, n]$.

▶ **Definition 34.** For all $A \in \binom{[n]}{x}$, $B \in \binom{[n]}{y}$, $C \in S_{n,z}$, define

$$Q(A,C,B) = \begin{cases} \chi_{H,A,B,C} & A \cap B = \emptyset, \\ & \forall i \in [1,x], a_i \in V_i, \forall j \in [1,y], b_j \in V_{x+j}, \forall k \in [1,z], c_k \in V_{x+y+k} \\ 0 & otherwise \end{cases}$$

Now note that

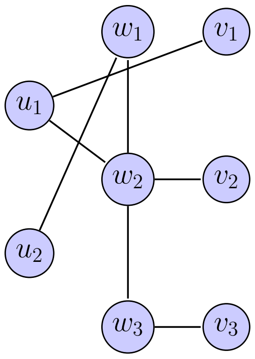
$$\begin{aligned}\mathbb{E}[\mathrm{tr}((R'R'^T)^k)] &= \sum_{\substack{A_1,A_5,\ldots,A_{4k-3}\in\binom{[n]}{x} \\ B_3,B_7,\ldots,B_{4k-1}\in\binom{[n]}{y}}} \mathbb{E}\left[\prod_{j=1}^{k} R'(A_{4j-3},B_{4j-1})R'^T(B_{4j-1},A_{4j+1})\right] \\ &= \sum_{\substack{A_1,A_5,\ldots,A_{4k-3}\in\binom{[n]}{x} \\ B_3,B_7,\ldots,B_{4k-1}\in\binom{[n]}{y} \\ C_2,C_4,\ldots C_{4k}\in S_{n,z}}} \mathbb{E}\left[\prod_{j=1}^{k} Q(A_{4j-3},C_{4j-2},B_{4j-1})Q^T(B_{4j-1},C_{4j},A_{4j+1})\right]\end{aligned}$$

by linearity of expectation, where $A_{4k+1} = A_1$.

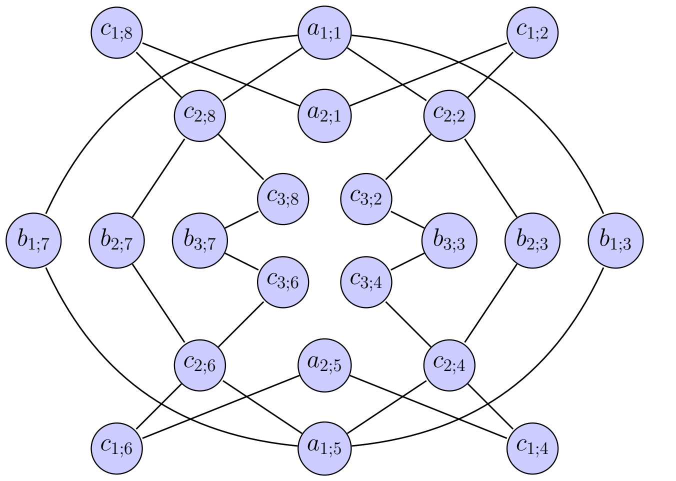
Denote $\prod_{j=1}^{k} Q(A_{4j-3},C_{4j-2},B_{4j-1})Q^T(B_{4j-1},C_{4j},A_{4j+1})$ as $P(A_1,C_2,\ldots,B_{4k-1},C_{4k})$. Similar to before, because $\mathbb{E}[Q(A,C,B)] = 0$ for randomly chosen A, B, and C, the vast majority of the terms $\mathbb{E}[P(A_1,C_2,\ldots,B_{4k-1},C_{4k})]$ are 0; in fact, the only time the expected value can be non-zero is when each consecutive pair of sets of variables is disjoint and every edge of G involved in the product appears an even number of times, in which case the expected value is at most 1. Again, we can upper bound the number of choices for $A_1,C_2,\ldots,B_{4k-1},C_{4k}$ that yield a non-zero value for $\mathbb{E}[P(A_1,C_2,\ldots,B_{4k-1},C_{4k})]$ and use that number to bound $\mathbb{E}[\mathrm{tr}((M'M'^T)^k)]$. In order to represent $\mathbb{E}[P(A_1,C_2,\ldots,B_{4k-1},C_{4k})]$, we use another constraint graph. This constraint graph is similar to the earlier one; however, it is slightly more complicated, as there is an extra set of vertices involved.

In this constraint graph, there are $k(x+2z+y)$ vertices sorted into $4k$ sets. These vertices are labeled $A_1 = \{a_{1;1},a_{2;1},\ldots,a_{x;1}\}, C_2 = \{c_{1;2},c_{2;2},\ldots,c_{z;2}\}, B_3 = \{b_{1;3},b_{2;3},\ldots,b_{y;3}\}$, $C_4 = \{c_{1;4},c_{2;4},\ldots,c_{z;4}\}, A_5 = \{a_{1;5},a_{2;5},\ldots,a_{x;5}\},\ldots,C_{4k} = \{c_{1;4k},c_{2;4k}\ldots,c_{z;4k}\}$. Note that, in particular, the sets describing the sets of vertices are labeled A_1, C_2, B_3, C_4,…, and repeat this pattern exactly k times. Two vertices $a_{p;q}$ and $b_{r;s}$ are adjacent if and only if $|q-s| = 2$ and u_p and v_r are adjacent in H, where $a_{p;1} = a_{p;4k+1}$. Similarly, $a_{p;q}$ and $c_{t;o}$ are adjacent if and only if $|q-o| = 1$ and u_p and w_t are adjacent in H, and $b_{r;s}$ and $c_{t;o}$ are adjacent if and only if $|s-o| = 1$ and v_r and w_t are adjacent in H. Finally, $c_{t;o}$ and $c_{t';o'}$ are adjacent if and only if $o = o'$ and w_t and $w_{t'}$ are adjacent in H.

Now, in order to bound $\mathbb{E}[\mathrm{tr}((R'R'^T)^k)]$, we must calculate the minimum number of constraint edges required in our constraint graph to bring about a non-zero expectation value, as that will help bound the number of choices for $A_1,C_2,B_3,C_4,A_5,\ldots,B_{4k-1},C_{4k}$ that yield a non-zero expectation value for the product $\mathbb{E}[P(A_1,C_2,\ldots,B_{4k-1},C_{4k})]$.



(a) H.



(b) An example of the constraint graph for the given example of H, where $k = 2$.

Figure 4

▶ **Lemma 35.** *In order for $\mathbb{E}[P(A_1, C_2, \dots, B_{4k-1}, C_{4k})]$ to have a non-zero value, there must be at least $q(k-1) + zk$ constraint edges in the respective constraint graph, where q is the maximal number of vertex-independent paths from X to Y in H. In addition, this bound is sharp.*

Proof. Choose q vertex-independent paths from U to V. Let l_i be the length of the ith such path.

Note that any path of length l_i in H from U to V corresponds to a cycle of size $2kl_i$ in our constraint graph; this cycle requires $kl_i - 1$ constraint edges by Proposition 17. Since the cycles are disjoint and by the definition of R' we cannot have constraint edges between them, the total number of constraint edges required by just the q vertex-independent paths is $\sum_{i=1}^{q}(kl_i - 1) = k(\sum_{i=1}^{q} l_i) - q$.

Consider the vertices in W which are not in the q paths. Of the z vertices in W, because each pair of paths is disjoint and a path of length l_i corresponds to exactly $l_i - 1$ vertices in C, exactly $\sum_{i=1}^{q}(l_i - 1) = (\sum_{i=1}^{q} l_i) - q$ vertices of C are included in paths, so there are $z - ((\sum_{i=1}^{q} l_i) - q)$ vertices of W not included in the paths. Each of these vertices is incident with an edge in H, and that edge is repeated $2k$ times in the constraint graph; therefore, as each edge of G that appears in the constraint graph must appear an even number of times in the constraint graph, any particular vertex of positive degree can, in all of its appearances in the constraint graph, only take on at most k values. However, the particular vertex appears $2k$ times in the constraint graph, once in each set of vertices of the form C_i, so it must have at least k constraint edges between those $2k$ appearances. Therefore, there are required to be a minimum of $(z - ((\sum_{i=1}^{q} l_i) - q))k + (k(\sum_{i=1}^{q} l_i) - q) = q(k-1) + zk$ constraint edges in the constraint graph.

In order to prove the sharpness, we utilize Menger's Theorem [21]. First, note that the existence of q vertex-independent paths from U to V implies that there exists q vertex-independent paths from U to V, with first vertex in U, last vertex in V, and all internal vertices in W. This statement is true because given these q vertex-independent paths, if any of them have internal vertices in U or V, we can simply shorten these paths until they have only first and last vertices in U and V.

Now, Menger's Theorem states that because there are a maximum of q vertex-indepedent paths from U to V in H with all internal vertices in W, there is a set $S \in V(H)$ with $|S| = q$ such that all paths from U to V pass through at least one vertex of S. Consider such a set S. Using S, we will create $q(k-1) + zk$ constraint edges that yield a non-zero expectation value for $\mathbb{E}[P(A_1, C_2, \dots, B_{4k-1}, C_{4k})]$ as follows.

For all vertices $u_i \in X$ such that $u_i \in S$, set $a_{i;1} = a_{i;5} = \cdots = a_{i;4k-3}$. Note that this requires $k-1$ constraint edges per vertex in S. Similarly, for all vertices $v_i \in Y$ such that $v_i \in S$, set $b_{i;3} = b_{i;7} = \cdots = b_{i;4k-1}$. In addition, for all vertices $w_i \in Z$ such that $w_i \in S$, set $c_{i;2} = c_{i;4} = \cdots = c_{i;4k}$. This requires $2k-1$ constraint edges per vertex in S.

Now, consider all vertices $w_i \in Z$ such that $w_i \notin S$. Note that if there existed a path from w_i to U that passed through no vertices in S, there cannot exist a path from w_i to V that passes through no vertices in S, as that would imply that there was a path from U to V not passing through any vertices on S. So, if there exists a path from w_i to U passing through no vertices of S, set $c_{i;4} = c_{i;6}, c_{i;8} = c_{i;10}, \dots, c_{i;4k} = c_{i;2}$. Otherwise, set $c_{i;2} = c_{i;4}, c_{i;6} = c_{i;8}, \dots, c_{i;4k-2} = c_{i;4k}$. This requires k constraint edges per vertex.

Now given an edge in the constraint graph $(a_{p;q}, b_{r;s})$ with $|q - s| = 2$, then either u_p or v_r is in S or else there would exist a path from U to V not in S; therefore, either $(a_{p;q}, b_{r;q+2}) = (a_{p;q}, b_{r;q-2})$ or $(a_{p;s-2}, b_{r;s}) = (a_{p;s+2}, b_{r;s})$ by the constraint edges. Similarly, given $(a_{p;q}, c_{t;o})$ with $|p - o| = 1$, either $c_{t;q-1} = c_{t;q+1}$, which implies $(a_{p;q}, c_{t;q-1}) = (a_{p;q}, c_{t;q+1})$, or $w_t \notin S$ and $u_p \in S$, which implies $(a_{p;2o-q-2}, c_{t;2o-q-1}) = (a_{p;2o-q+2}, c_{t;2o-q+1})$. A similar argument applies to edges of the form $(c_{t;o}, b_{r;s})$. For edges of the form $(c_{t;o}, c_{t';o})$, it can be shown that either $(c_{t;o}, c_{t',o}) = (c_{t;o-2}, c_{t';o-2})$ or $(c_{t;o}, c_{t',o}) = (c_{t;o+2}, c_{t',o+2})$. Finally, note that edges in the constraint graph of the form $(a_{p_1;q}, a_{p_2;q})$ and $(b_{r_1;s}, b_{r_2;s})$ are automatically doubled and have no effect. Thus, this construction makes every edge appear with an even multiplicity, as needed.

If S contains exactly j vertices in W, then the total number of constraint edges used in this construction is $(k-1)(|S| - j) + (2k-1)j + (k)(z-j) = q(k-1) + zk$, meaning that the bound given is sharp. ◀

▶ **Corollary 36.** *Let N represent the number of choices for A_1, C_2, ..., B_{4k-1}, B_{4k} such that*
$\mathbb{E}[P(A_1, C_2, \dots, B_{4k-1}, C_{4k})] \neq 0$. *Then,* $N \leq ((t+z)k)^{2k(z+q)-2q} n^{(t-q)k+q}$.

Proof. Apply Proposition 18. In this situation, $b = k(t+z)$ and $c = q(k-1) + zk$. This implies the desired result. ◀

▶ **Corollary 37.** $\mathbb{E}[\mathrm{tr}((R'R'^T)^k)] \leq ((t+z)k)^{2k(z+q)-2q} n^{(t-q)k+q}$.

Proof. Recall that

$$\mathbb{E}[\mathrm{tr}((R'R'^T)^k)] = \sum_{\substack{A_1, A_5, \dots, A_{4k-3} \in S_{n,u} \\ B_3, B_7, \dots, B_{4k-1} \in S_{n,v} \\ C_2, C_4, \dots C_{4k} \in S'_{n,w}}} \mathbb{E}\left[\prod_{j=1}^{k} Q(A_{4j-3}, C_{4j-2}, B_{4j-1}) Q^T(B_{4j-1}, C_{4j}, A_{4j+1})\right].$$

Then, by Proposition 36, there are at most $((t+z)k)^{2k(z+q)-2q} n^{(t-q)k+q}$ choices for $A_1, C_2, \dots, B_{4k-1}, C_{4k}$ that yield a non-zero value for

$\mathbb{E}\left[\prod_{j=1}^{k} Q(A_{4j-3}, C_{4j-2}, B_{4j-1})Q^T(B_{4j-1}, C_{4j}, A_{4j+1})\right]$; in addition, $\mathbb{E}\left[\prod_{j=1}^{k} Q(A_{4j-3}, C_{4j-2}, B_{4j-1})Q^T(B_{4j-1}, C_{4j}, A_{4j+1})\right] \leq 1$. These two observations complete the proof. ◀

Now for any graph G on n vertices, $\mathrm{tr}((R'R'^T)^k)$ must take on a nonnegative value. By Markov's inequality and Corollary 37, for all $\epsilon \in (0,1)$

$$\mathbb{P}\left[\mathrm{tr}((R'R'^T)^k) \geq \frac{\mathbb{E}[\mathrm{tr}((R'R'^T)^k)]}{\epsilon}\right] \leq \mathbb{P}\left[\mathrm{tr}((R'R'^T)^k) \geq ((t+z)k)^{2k(z+q)-2q} n^{(t-q)k+q}/\epsilon\right] \leq \epsilon.$$

Since $||R'|| \leq \sqrt[2k]{\mathrm{tr}((R'R'^T)^k)}$, this implies that for all $k \geq 1$ and all $\epsilon \in (0,1)$,

$$\mathbb{P}\left[||R'|| \geq \sqrt[2k]{((t+z)k)^{2k(z+q)-2q} n^{(t-q)k+q}/\epsilon}\right] \leq \mathbb{P}\left[||R'|| \geq ((t+z)k)^{z+q} n^{\frac{t-q}{2}} (n^q/\epsilon)^{1/2k}\right] \leq \epsilon.$$

Setting $k = \lceil \frac{1}{2(q+z)} \ln(n^q/\epsilon) \rceil$ we have that

$$((t+z)k)^q n^{\frac{t-q}{2}} (n^q/\epsilon)^{1/2k} \leq \left((t+z)\left(\frac{\ln(n^q/\epsilon)}{2(q+z)} + 1\right)\right)^{q+z} n^{\frac{t-q}{2}} e^{q+z}.$$

Therefore, $\mathbb{P}[||R'|| \geq \left(e(t+z)\left(\frac{\ln(n^q/\epsilon)}{2(q+z)} + 1\right)\right)^{q+z} n^{\frac{t-q}{2}}] \leq \epsilon$, as needed. ◀

Using Lemma 27 with $p = \epsilon$ and $B = 2(t^t)\left(e(t+z)\left(\frac{\ln(8n^q/\epsilon)}{2(q+z)} + 1\right)\right)^{q+z} n^{\frac{t-q}{2}}$ and following the same logic as in the proof of Theorem 21, we obtain that for all $\epsilon \in (0,1)$,

$$\mathbb{P}[||R_H|| \geq 2(t^t)\left(e(t+z)\left(\frac{\ln(8n^q/\epsilon)}{2(q+z)} + 1\right)\right)^{q+z} n^{\frac{t-q}{2}}] \leq \epsilon\,.$$ ◀

6 Lower Bounds

In this section, we show that the bounds we have obtained on the norms of uniform low degree graph matrices are tight up to a factor of polylog(n). This makes intuitive sense as our bounds on $E[tr((R'R'^T)^k)]$ were tight up to a polylog factor. Unfortunately, these lower bounds on $E[tr((R'R'^T)^k)]$ are insufficient for two reasons. First, they do not rule out the possibility that $||R'||$ is sometimes very small. Second, lower bounds on the norms of the matrices R' do not imply a lower bound on $||R_H||$. We now show how these obstacles can be overcome (though we only give a proof sketch as a full proof would be long and technical).

▶ **Theorem 38.** *Let H be a graph with distinguished sets of vertices U and V where U and V are disjoint and for all vertices w in $V(H) \setminus (U \cup V)$, there is a path from w to either U or V in H. Letting $t = |V(H)|$ and letting q be the minimal size of a vertex separator between U and V, with high probability $||R_H||$ is $\Omega(n^{\frac{t-q}{2}})$.*

▶ Remark. If H has non-isolated vertices which are not connected to U or V, there is a non-negligible chance that R_H has considerably smaller norm than expected. To see this, let H_0 be the part of H which is connected to U and/or V and let H_1 be the remainder of H. We have that $R_H \approx R_{H_1}R_{H_0}$ where R_{H_1} is a constant depending on the input graph G which has some chance of being close to 0.

Before giving a proof sketch for Theorem 38, we first consider the case when H is bipartite with partite sets U and V, which can be analyzed directly.

▶ **Theorem 39.** *Let H be a bipartite graph with partite sets U and V. Letting $t = |V(H)|$ and letting q be the minimal size of a vertex cover of H, $||R_H||$ is $\Omega(n^{\frac{t-q}{2}})$.*

Proof. We show this by finding vectors u and v such that $u^T R_H v$ is $\Theta(n^{\frac{t-q}{2}})||u|| \cdot ||v||$. The idea is to take a minimal vertex cover S of U and V and fix the vertices that S maps to. This essentially separates the dependence of $R_H(A, B)$ on A and B which means that we can choose u to match the dependence on A and choose v to match the dependence on B.

▶ **Definition 40.**
1. Define $E_L \subseteq E(H)$ to be the edges in H between $U \setminus S$ and $S \cap V$.
2. Define $E_M \subseteq E(H)$ to be the edges in H between $S \cap U$ and $S \cap V$
3. Define $E_R \subseteq E(H)$ to be the edges in H between $S \cap U$ and $V \setminus S$.

Now choose disjoint vertices $A_S \cup B_S$ for the vertices of S to map into and define u and v as follows.

▶ **Definition 41.** Given an A which is disjoint from B_S, letting π be the map which maps U into A and maps $S \cap V$ into B_S, define u_A to be $\chi_{\pi(E_L)}$ if $\pi(S \cap U) = A_S$ and 0 otherwise.

Given a B which is disjoint from A_S, letting π be the map which maps V into B and maps $S \cap U$ into A_S, define v_B to be $\chi_{\pi(E_R)}$ if $\pi(S \cap V) = B_S$ and 0 otherwise.

Note that $u_A R_H(A, B) v_B$ is only nonzero if the following conditions hold
1. A and B are disjoint
2. If π is the map that maps U into A and V into B then $\pi(S \cap U) = A_S$ and $\pi(S \cap V) = B_S$.

When these conditions hold, $u_A R_H(A, B) v_B = \chi_{\pi(E_L)} \chi_{\pi(E(H))} \chi_{\pi(E_R)} = \chi_{\pi(E_M)}$ which will always be the same as A_S and B_S are fixed. There are $\Theta(n^{|U|-|S \cap U|})$ A such that $u_A \neq 0$, there are $\Theta(n^{|V|-|S \cap V|})$ B such that $v_B \neq 0$, and there are $\Theta(n^{|U|-|S \cap U|} n^{|V|-|S \cap V|}) = \Theta(n^{t-q})$ choices for A and B for which these conditions hold. This implies that $||u||$ is $\Theta(n^{\frac{|U|-|S \cap U|}{2}})$, $||v||$ is $\Theta(n^{\frac{|V|-|S \cap V|}{2}})$, and $|u^T R_H v|$ is $\Theta(n^{t-q})$. Putting everything together, $|u^T R_H v|$ is $\Theta(n^{\frac{t-q}{2}})||u|| \cdot ||v||$, so $||R_H||$ is $\Omega(n^{\frac{t-q}{2}})$, as needed. ◀

In the general case, we could use similar ideas, choosing a minimal vertex seperator S of U and V in H, fixing the vertices that S maps to, and then choosing u to match the dependence on A and v to match the dependence on B. However, the analysis is tricky for two reasons. First, it is non-trivial to bound $||u||$ and $||v||$ as the entries of u and v are the sums of many terms. Second, $u^T R_H v$ will have additional terms coming from different choices for the vertices in $S \setminus (U \cup V)$ in the sums describing the entries of R_H. To deal with these issues, we use an argument involving $||R_H||_{Fr}$, the Frobenius norm of R_H, which is somewhat cleaner to analyze.

Proof Sketch of Theorem 38.

▶ **Definition 42.** Given two matrices M_1 and M_2 with the same dimensions, we define $\langle M_1, M_2 \rangle = \sum_{i,j} M_1(i, j) M_2(i, j)$. Note that for any matrix M, $\langle M, M \rangle = ||M||_{Fr}^2$

Given a matrix M, we can bound $||M||$ as follows.

▶ **Proposition 43.** *If $M = \sum_i c_i u_i v_i^T$ for some vectors u_i, v_i and some positive coefficients c_i then $||M|| \geq \frac{||M||_{Fr}^2}{\sum_i c_i ||u_i|| \cdot ||v_i||}$*

Proof. Note that

$$||M||_{Fr}^2 = \langle M, M\rangle = \langle M, \sum_i c_i u_i v_i^T\rangle = \sum_i c_i u_i^T M v_i \leq ||M|| \sum_i c_i ||u_i|| \cdot ||v_i|| \qquad ◀$$

With this proposition in mind, we first describe how we can decompose R_H. We then describe how to bound $||R_H||_{Fr}^2$ and the norms of the vectors involved.

▶ **Definition 44.** Given a graph H where all vertices are connected to U or V and a minmal separator S of U and V,

1. Let L be the set of vertices which are connected to U once we remove S from H
2. Let R be set of vertices connected to V once we remove S from H.
3. Let H_L be the graph with vertices $L \cup S$ and all edges in H which are incident with a vertex in L.
4. Let H_R be the graph with vertices $R \cup S$ and all edges in H which are between two vertices in S or are incident with a vertex in R.
5. Let $l = |L|$, $q = |S|$, and $r = |R|$. Note that $t = l + q + r$.

▶ **Proposition 45.**

$$R_{H,V_1,\ldots,V_t} = \sum_{\substack{v_{l+1},\ldots,v_{l+q}: \\ \forall i \in [l+1,l+q], v_i \in V_i}} R_{H_L,V_1,\ldots,V_l,\{v_{l+1}\},\ldots,\{v_{l+q}\}} R_{H_R,\{v_{l+1}\},\ldots,\{v_{l+q}\},V_{l+q+1},\ldots,V_t}$$

where we take $V = \emptyset$ for H_L and we take $U = \emptyset$ for H_R (so $R_{H_L,V_1,\ldots,V_l,\{v_{l+1}\},\ldots,\{v_{l+q}\}}$ and $R^T_{H_R,\{v_{l+1}\},\ldots,\{v_{l+q}\},V_{l+q+1},\ldots,V_t}$ are in fact vectors)

Proof. This proposition follows directly from the definitions of the matrices involved. ◀

Writing $R_H = \frac{t^t}{t^n} \sum_{V_1,\ldots,V_t} R_{H,V_1,\ldots,V_t}$, we have that $R_H = \sum_{c_i} u_i v_i^T$ where each vector u_i is of the form $R_{H_L,V_1,\ldots,V_l,\{v_{l+1}\},\ldots,\{v_{l+q}\}}$ and each vector v_i is of the form $R^T_{H_R,\{v_{l+1}\},\ldots,\{v_{l+q}\},V_{l+q+1},\ldots,V_t}$. Note that the sum of the coefficients of the vector products in this sum is $O(n^q)$. We now probabilistically bound $||R_H||_{Fr}^2$, $||u_i||$, and $||v_i||$.

▶ **Lemma 46.** *Let H_1 and H_2 be two graphs such that both H_1 and H_2 have distinguished sets of vertices U and V, U and V are disjoint, every vertex in H_1 is connected to a vertex in U or V, and every vertex in H_2 is connected to a vertex in U or V. Let $t_1 = |V(H_1)|$ and let $t_2 = |V(H_2)|$.*

1. *If $H_1 = H_2 = H$ (after permuting the vertices not in $U \cup V$) then letting $t = t_1 = t_2$, $E[\langle R_H, R_H\rangle]$ is $\Theta(n^t)$ and with high probability, $\langle R_H, R_H\rangle - E[\langle R_H, R_H\rangle]$ is $O(n^{t-\frac{1}{2}} polylog(n))$*
2. *If H_1 and H_2 are different graphs then with high probability, $\langle R_{H_1}, R_{H_2}\rangle$ is $O(n^{\frac{t_1+t_2-1}{2}} polylog(n))$*

Proof. Consider the terms in $\langle R_{H_1}, R_{H_2}\rangle$. Each such term is determined by mappings $\pi_1 : V(H_1) \to G$, $\pi_2 : V(H_2) \to G$ where $\pi_2(U) = \pi_1(U)$, $\pi_2(V) = \pi_1(V)$, and these maps preserve the ordering of U and V. For a given term, let H' be the graph with vertices $\pi_1(V(H_1)) \cup \pi_2(V(H_2))$ and edges $\pi_1(E(H_1)) \Delta \pi_2(E(H_2))$ where Δ is the symmetric difference. If we group the terms with the same graph H' (up to a permutation of the vertices) together, then we obtain sums of the following form.

▶ **Definition 47.** Given a graph H' with two distinguished subsets of vertices U and V, define $f_{H'}(G) = \sum_{A,B} R_{H'}(A, B)$

We have the following probabilistic bound on these sums.

▶ **Lemma 48.** *If x is the number of non-isolated vertices in H' and y is the number of isolated vertices in H', with high probability $f_{H'}(G)$ is $O(n^{\frac{x}{2}+y}polylog(n))$*

Proof Sketch. We show the result for a related function $f_{H',V_1,\ldots,V_m}(G)$ where we restrict where the vertices of H' map to. To rigorously show the result, we would then use Lemma 27 (which applies just as well to scalars).

▶ **Definition 49.** Letting $t' = |V(H')|$ and given disjoint sets of vertices $V_1, \ldots, V_m$, define $f_{H',V_1,\ldots,V_{t'}}(G) = \sum_{A,B} R_{H',V_1,\ldots,V_{t'}}(A, B)$

▶ **Lemma 50.** *If x be the number of non-isolated vertices in H' and y is the number of isolated vertices in H;, with high probability $f_{H',V_1,\ldots,V_m}(G)$ is $O(n^{\frac{x}{2}+y})$*

Each term in $f_{H',V_1,\ldots,V_m}(G)$ can be described by choosing a vertex v_i from each V_i. To prove this result, we start by considering each term individually, thinking of all the vertices v_i as being fixed. We then iteratively group these terms together by choosing one or two vertices which are still fixed and summing over all possibilities for these vertices.

When we sum over the possibilities for some vertex v_i, we randomly fix all of the edges of G which are incident with a vertex in V_i and call v_i free. By doing this, we always know the magnitudes of all our sums (but not their signs!). At each point, we have the following bound.

▶ **Lemma 51.** *We can choose the order in which we make vertices free so that if we have made x_1 non-isolated vertices free and y_1 isolated vertices free then with high probability our sums are all $O(n^{\frac{x_1}{2}+y_1}polylog(n))$. Moreover, at all times, for every non-isolated vertex v_i, there is an edge between v_i and another fixed vertex v_j in $\pi(E(H))$ (where π is the mapping from $V(H)$ to $V(G)$).*

Proof. We prove this result by induction. The base case $x_1 = y_1 = 0$ is trivial. If there is an isolated vertex which is fixed, there are at most n possibilities for this vertex, so this grouping increases y_1 by 1 and multiplies our bound by a factor of at most $O(n)$. If there is a non-isolated fixed vertex v_i which can be made free while maintaining the invariant that every fixed vertex has an edge to another fixed vertex in $\pi(E(H))$, sum over the possibilities for this vertex. In this sum, we know the magnitude of each term, but the signs of each term are random and completely independent from each other (as they depend on edge(s) between the different possibilities for v_i and other fixed vertices). This summation increases x_1 by 1 and since the signs are independent, with high probability this summation increases our bound by a factor of at most $O(\sqrt{n}log(n))$. The remaining case is when $\pi(E(H))$ contains a perfect matching between the fixed vertices and no other edges between fixed vertices. In this case, choose one such edge (v_i, v_j) and sum up over the possibilities for v_i and v_j. Again, we know the magnitudes of each term in this sum, but the signs of each term are independent (as they depend on the different edges (v_i, v_j)). This summation increases x_1 by 2 and since the signs are independent, with high probability this summation increases our bound by a factor of at most $O(nlog(n))$. ◀

▶ Remark. The reason we needed to use $f_{H',V_1,\ldots,V_{t'}}(G)$ rather than $f_{H'}(G)$ in this argument is that it allowed us to consistently choose which edges of G we fixed and which edges of G were still undetermined at any given point. ◀

We now bound the terms in $\langle R_{H_1}, R_{H_2}\rangle$ using Lemma 48. We consider all of the terms $f_{H'}$ which appear in this sum. If $\pi_1(H_1) = \pi_2(H_2)$ then H' consists of t_1 isolated vertices and we can see directly that $f_{H'}(G)$ is $\Theta(n^{t_1})$.

For a given $f_{H'}$ such that $\pi_1(H_1) \neq \pi_2(H_2)$, let $r = |\pi_1(V(H_1)) \cap \pi_2(V(H_2))|$. Letting x be the number of non-isolated vertices in H' and y be the number of isolated vertices in H', we have that $x + y = |V(H')| = t_1 + t_2 - |U| - |V| - r$. Note that only the vertices in $\pi_1(U) \cup \pi_1(V) \cup \pi_1(V(H_1)) \cap \pi_2(V(H_2))$ can be isolated in H' because all other vertices are incident with an edge in $\pi_1(E(H_1)) \cup \pi_2(E(H_2))$ which only appears once. This tells us that $y \leq |U| + |V| + r$. Applying Lemma 48, we have that $f_{H'}$ is $O(n^{\frac{x}{2}+y} polylog(n)) = O(n^{\frac{(x+y)+y}{2}} polylog(n))$ which is $O(n^{\frac{t_1+t_2}{2}} polylog(n))$.

To improve this bound and obtain our result, it is sufficient to show that if $\pi_1(H_1) \neq \pi_2(H_2)$, there must be some vertex in $\pi_1(U) \cup \pi_1(V) \cup \pi_1(V(H_1)) \cap \pi_2(V(H_2))$ which is not isolated, as this reduces our upper bound on y by 1. To show this, first note that if $\pi_1(V(H_1)) = \pi_2(V(H_2))$ yet $\pi_1(H_1) \neq \pi_2(H_2)$ then H' must have an edge so not all of the vertices of H' can be isolated. If $\pi_1(V(H_1)) \neq \pi_2(V(H_2))$ then either $\pi_1(V(H_1)) \setminus \pi_2(V(H_2))$ is nonempty or $\pi_2(V(H_2)) \setminus \pi_1(V(H_1))$ is nonempty. Without loss of generality, we may assume that $\pi_1(V(H_1)) \setminus \pi_2(V(H_2))$ is nonempty. Let v_i be a vertex in $\pi_1(V(H_1)) \setminus \pi_2(V(H_2))$. Since every vertex in H_1 is connected to either U or V, if we look at the edges $\pi_1(E(H_1)) \cup \pi_2(E(H_2))$, there must be a path from v_i to some vertex v_j in $\pi_1(U \cup V)$. There must be an edge in this path between a vertex in $\pi_1(V(H_1)) \setminus \pi_2(V(H_2))$ and a vertex in $\pi_1(U) \cup \pi_1(V) \cup \pi_1(V(H_1)) \cap \pi_2(V(H_2))$, let (v_k, v_l) be the first such edge. $v_l \in \pi_1(U) \cup \pi_1(V) \cup \pi_1(V(H_1)) \cap \pi_2(V(H_2))$ and cannot be isolated in H', so the result follows. ◀

We now give a sketch for how to probabilistically bound the norms of the vectors u_i and the vectors v_i and complete the proof. Recall that each vector u_i is of the form $R_{H_L,V_1,\ldots,V_{l+q}}$ where the V for H_L is empty and $V_{l+1}, \ldots, V_{l+q}$ have size 1. We now use similar reasoning as we used to prove Lemma 46 except that there is only one possibility for the vertices corresponding to $V_{l+1}, \ldots, V_{l+q}$ so we always keep these vertices fixed (and don't sum over their possibilities). Applying this reasoning, we obtain that for each i, with high probability, $||u_i||^2$ is $\Theta(n^l)$ so $||u_i||$ is $\Theta(n^{\frac{l}{2}})$. By symmetry, for each i, with high probability $||v_i||$ is $\Theta(n^{\frac{r}{w}})$. Putting everything together, with high probability $||R_H||_{Fr}^2$ is $\Theta(n^t)$ and $\sum_i c_i ||u_i|| \cdot ||v_i||$ is $O(n^{\frac{l+r}{2}+q}) = O(n^{\frac{t+s}{2}})$. $||R_H|| \geq \frac{||R_H||_{Fr}^2}{\sum_i c_i ||u_i|| \cdot ||v_i||}$, which is $\Omega(n^{\frac{t-q}{2}})$, as needed. ◀

▶ Remark. While Theorem 38 was only stated for a single R_H, the techniques used to prove Theorem 38 apply just as well to a linear combination of such matrices. In particular, if we have distinct graphs $H_1, \ldots, H_k$ which all satisfy the conditions of Theorem 38 then for any coefficients $c_1, \ldots, c_k$, with high probability, $|| \sum_{i=1}^{k} c_k R_{H_k} ||$ is $\Theta(\max_i\{||c_i R_{H_i}||\})$

7 Conclusion and Further Studies

In this paper, we analyzed the norms of uniform low degree graph matrices, which appear naturally when analyzing the sum of squares hierarchy. While special cases of these matrices were analyzed in previous works on sum of squares lower bounds for the planted clique problem [20], we generalized this analysis, proving an upper bound on the norms of all such matrices which is tight up to a polylogarithmic factor. This general analysis is a key component of the work [4] proving almost tight lower bounds for sum of squares on planted clique and will very likely be useful for further analysis of the sum of squares hierarchy.

That said, there are several open problems raised by this work. First, to what extent can these norm bounds be improved? It is very likely that with a more careful analysis, the polylog factors can be reduced or removed and the dependence of $||R_H||$ on the size of the graph H can be improved. Can we go further and determine the distribution of these matrices' eigenvalues? Second, what can we say about the non-uniform case? How much structure do we need our matrices to have to obtain interesting norm bounds?

Acknowledgements. This research was supported by the Program for Research in Mathematics, Engineering, and Science (PRIMES-USA) at MIT. We thank the PRIMES-USA faculty, including Dr. Tanya Khovanova, Dr. Pavel Etingof, and Dr. Slava Gerovitch for helpful feedback on this work.

References

1 Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. *Random Structures and Algorithms*, 13(3-4):457–466, 1998.

2 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 563–572. IEEE, 2010.

3 Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5, 2009.

4 Boaz Barak, Sam Hopkins, Jonathan Kelner, Ankur Moitra, Pravesh Kothari, and Aaron Potechin. A Nearly Tight Sum-of-Squares Lower Bound for the Planted Clique Problem. *ArXiv e-prints*, April 2016. `arXiv:1503.06447`.

5 Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 472–481. IEEE, 2011.

6 Boaz Barak and David Steurer. Sum-of-squares proofs and the quest toward optimal algorithms. In *Proceedings of International Congress of Mathematicians (ICM)*, 2014.

7 Yash Deshpande and Andrea Montanari. Improved sum-of-squares lower bounds for hidden clique and hidden submatrix problems. *CoRR*, abs/1502.06590, 2015. URL: `http://arxiv.org/abs/1502.06590`.

8 Uriel Feige and Robert Krauthgamer. The probable value of the lovász–schrijver relaxations for maximum independent set. *SIAM J. Comput.*, 32(2):345–370, 2003. `doi:10.1137/S009753970240118X`.

9 Vitaly Feldman, Elena Grigorescu, Lev Reyzin, Santosh Vempala, and Ying Xiao. Statistical algorithms and a lower bound for detecting planted cliques. In *Proceedings of the forty-fourth annual ACM symposium on Theory of Computing*. ACM, 2013.

10 Vyacheslav L. Girko. Circular law. *Theory of Probability and its Applications*, 29:694–706, 1984.

11 Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

12 Dima Grigoriev. Complexity of positivstellensatz proofs for the knapsack. *Computational Complexity*, 10(2):139–154, 2001.

13 Dima Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theoretical Computer Science*, 259(1):613–622, 2001.

14 Venkatesan Guruswami and Ali Kemal Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with psd objectives. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 482–491. IEEE, 2011.

15 Samuel B. Hopkins, Pravesh K. Kothari, and Aaron Potechin. Sos and planted clique: Tight analysis of MPW moments at all degrees and an optimal lower bound at degree four. *CoRR*, abs/1507.05230, 2015. URL: `http://arxiv.org/abs/1507.05230`.

16 Mark Jerrum. Large cliques elude the metropolis process. *Random Structures & Algorithms*, 3(4):347–359, 1992.

17 Denés Konig. Gráfok és mátrixok. matematikai és fizikai lapok, 38: 116–119, 1931.

18 Luděk Kučera. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics*, 57(2):193–212, 1995.

19 Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

20 R. Meka, A. Potechin, and A. Wigderson. Sum-of-squares lower bounds for planted clique. *ArXiv e-prints*, March 2015. `arXiv:1503.06447`.

21 Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

22 Yurii Nesterov. Squared functional systems and optimization problems. In *High performance optimization*, pages 405–440. Springer, 2000.

23 Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization.* PhD thesis, Citeseer, 2000.

24 Prasad Raghavendra and Tselil Schramm. Tight lower bounds for planted clique in the degree-4 SOS program. *CoRR*, abs/1507.05136, 2015. URL: `http://arxiv.org/abs/1507.05136`.

25 Grant Schoenebeck. Linear level lasserre lower bounds for certain k-csps. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 593–602. IEEE, 2008.

26 NZ Shor. Class of global minimum bounds of polynomial functions. *Cybernetics and Systems Analysis*, 23(6):731–734, 1987.

27 Eugene P Wigner. On the distribution of the roots of certain symmetric matrices. *Annals of Mathematics*, pages 325–327, 1958.

A Justification of the moment method

In this appendix, we provide a proof that the moment method gives an upper bound on the norm. To make the proof easier, we consider the case of positive semidefinite matrices separately.

▶ **Lemma 52.**
1. *For any positive semidefinite matrix A, for all $k \geq 1$, $\sqrt[k]{\mathrm{tr}(A^k)} \geq ||A||$.*
2. *For any real matrix M, for all $k \geq 1$, $\sqrt[2k]{\mathrm{tr}((MM^T)^k)} \geq ||M||$.*

Proof. To show the first statement, we recall the following fact about positive semidefinite matrices.

▶ **Proposition 53.** *For any positive semidefinite matrix A, $||A|| = \lambda_{max}(A)$ where $\lambda_{max}(A)$ is the maximum eigenvalue of A.*

Proof. Since A is positive semidefinite, there is an orthonomal basis $v_1, \cdots, v_n$ of eigenvectors of A with eigenvalues $\lambda_1, \cdots, \lambda_n$. Without loss of generality, we may assume that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Given a unit vector v, $v = \sum_{i=1}^{n} c_i v_i$ where $\sum_{i=1}^{n} c_i^2 = 1$. $Av = \sum_{i=1}^{n} \lambda_i c_i v_i$ so we have that

$$||Av||^2 \leq \sum_{i=1}^{n} \lambda_i^2 c_i^2 \leq \sum_{i=1}^{n} \lambda_1^2 c_i^2 = \lambda_1^2 .$$

This implies that $||A|| \leq \lambda_1$. $||Av_1|| = \lambda_1$ so $||A|| \geq \lambda_1$ and thus $||A|| = \lambda_1$, as needed. ◀

With this in mind, for any $k \geq 1$, the eigenvalues of A^k are $\lambda_1^k, \cdots, \lambda_n^k$. Thus,

$$trace(A^k) = \sum_{i=1}^{n} \lambda_i^k \geq \lambda_1^k \geq ||A||^k$$

and the first statement follows. The second statement follows immediately from the first statement and the following proposition.

▶ **Proposition 54.** *For any matrix M, $||MM^T|| = ||M||^2$.*

Proof. Note that for any matrix M, $||M^T|| = ||M|| = \max_{||u||=1,||v||=1} u^T M v$. For any unit vectors u and v,

$$u^T M M^T v = (M^T u)^T (M^T v) = M^T u \cdot M^T v \leq ||M^T u|| \cdot ||M^T v|| \leq ||M^T||^2 = ||M||^2 .$$

Thus, $||M^T M|| \leq ||M||^2$. On the other hand, letting v be a unit vector which maximizes $||M^T v||$, giving $||M^T v|| = ||M^T|| = ||M||$, we have that

$$v^T M M^T v = (M^T v)^T (M^T v) = ||M^T v||^2 = ||M^T||^2 = ||M||^2 .$$

Thus, $||M^T M|| \geq ||M||^2$ and the result follows. ◀ ◀

B Norm bounds with left/right intersections

In this appendix, we consider the case when $U \cap V$ is nonempty in H.

▶ **Theorem 55.** *Let H be a graph with distinguished sets of vertices U and V such that $|U \cap V| = r$ and all vertices in $H(V) \setminus (U \cup V)$ have degree at least one. Let $t = |V(H)|$, let $z = |V(H) \setminus (U \cup V)|$, and let q be the size of the minimal separator between U and V (which must include $U \cap V$). Let $t' = t - r$ and let $q' = q - r$.*

1. *If $q' + z > 0$ then for all $\epsilon \in (0, 1)$,*

$$\mathbb{P}[||R_H|| \geq 2(t')^{t'} \left(e(t'+z)\left(\frac{\ln(8n^{q'+r}/\epsilon)}{2(q'+z)} + 1\right)\right)^{q'+z} n^{\frac{t'-q'}{2}}] \leq \epsilon .$$

2. *If $q' = z = 0$ then $||R_H|| \leq n^{\frac{t}{2}}$.*

Proof Sketch. The key idea is to reduce the case when to the case when $U \cap V = \emptyset$. We first choose the elements which $U \cap V$ map to. These will be the elements of $A \cap B$ and they must occur in fixed positions within A and B. Thus, this choice partitions R_H into blocks which share no rows or columns. It is now sufficient to obtain a probabilistic bound for each block and use a union bound.

For a particular block, we can now use the same proof we used to prove Theorem 14. Let R' be a matrix where we have restricted ourselves to a particular block and have partitioned the vertices in $[1, n] \setminus (A \cap B)$ into $V_1, \ldots, V_{t'}$, restricting where the vertices in $V(H) \setminus (U \cap V)$ can map to accordingly. Consider $tr((R'R'^T)^k)$. Roughly speaking, we ignore the vertices in $A \cap B$ (in fact we could replace n by $n' = n - r$ in the bound). Since the vertices in $A \cap B$ are fixed, they appear in every copy of R' and are already determined, so they do not contribute to the number of terms with nonzero expectation.

For the other vertices, it is still true that if we have vertex-independent paths of lengths $l_1, \ldots, l_{q'}$ in H from $U \setminus V$ to $V \setminus U$, in the constraint graph the path of length l_i becomes a cycle of length $2kl_i$, requiring $kl_i - 1$ constraint edges. It is also still true that every vertex in W which is not on one of these paths requires k constraint edges. Thus, we have the same norm bound on each block as we did in Theorem 14. To take a union bound over all the blocks, since there are at most n^r blocks, we use this bound with ϵ replaced by $\frac{\epsilon}{n^r}$ and the result follows. ◀

Lower Bounds for CSP Refutation by SDP Hierarchies

Ryuhei Mori*[1] and David Witmer†[2]

1 **Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Japan**
`mori@is.titech.ac.jp`

2 **Computer Science Department, Carnegie Mellon University, Pittsburgh, USA**
`dwitmer@cs.cmu.edu`

Abstract

For a k-ary predicate P, a random instance of $\mathrm{CSP}(P)$ with n variables and m constraints is unsatisfiable with high probability when $m \geq O(n)$. The natural algorithmic task in this regime is *refutation*: finding a proof that a given random instance is unsatisfiable. Recent work of Allen et al. suggests that the difficulty of refuting $\mathrm{CSP}(P)$ using an SDP is determined by a parameter $\mathrm{cmplx}(P)$, the smallest t for which there does not exist a t-wise uniform distribution over satisfying assignments to P. In particular they show that random instances of $\mathrm{CSP}(P)$ with $m \gg n^{\mathrm{cmplx(P)}/2}$ can be refuted efficiently using an SDP.

In this work, we give evidence that $n^{\mathrm{cmplx}(P)/2}$ constraints are also *necessary* for refutation using SDPs. Specifically, we show that if P supports a $(t-1)$-wise uniform distribution over satisfying assignments, then the Sherali-Adams$_+$ and Lovász-Schrijver$_+$ SDP hierarchies cannot refute a random instance of $\mathrm{CSP}(P)$ in polynomial time for any $m \leq n^{t/2-\varepsilon}$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases constraint satisfaction problems, LP and SDP relaxations, average-case complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.41

1 Introduction

The average-case complexity of constraint satisfaction problems (CSPs) has been studied in computer science, mathematics, and statistical physics. Despite the vast amount of research that has been done, the hardness of natural algorithmic tasks for random CSPs remains poorly understood. We consider random CSPs with n variables and m constraints chosen independently and uniformly at random. Whether or not a random CSP is satisfiable depends on its clause density $\frac{m}{n}$. It is conjectured that for any nontrivial CSP there is a *satisfiability threshold* $\alpha(P)$ depending on the choice of predicate P: For $m < \alpha(P) \cdot n$, an instance is satisfiable with high probability, and $m > \alpha(P)$, an instance is unsatisfiable with high probability. This conjecture has been proven in the case of k-SAT for large enough k [16]. For an arbitrary predicate P, it is only known that there exist constants $\alpha_{lb}(P)$ and $\alpha_{ub}(P)$ such that random instances with $m < \alpha_{lb}(P) \cdot n$ are satisfiable with high probability and random instances with $m > \alpha_{ub}(P) \cdot n$ are unsatisfiable with high probability. In the

* Supported by MEXT KAKENHI Grant Number 24106008.

† Supported by the National Science Foundation Graduate Research Fellowship Program under grant DGE-1252522, by NSF grants CCF-0747250 and CCF-1116594, and by a CMU Presidential Fellowship.

low density, satisfiable regime, the major research goal is to develop algorithms that find satisfying assignments. In the high density, unsatisfiable regime, the goal is to *refute* an instance, i.e., find a short certificate that there is no solution.

In this paper, we study refutation. A refutation algorithm takes a random instance $\mathcal{I}$ of $\mathrm{CSP}(P)$ and returns either “unsatisfiable" or "don't know". It must satisfy two conditions: (1) it is never wrong, i.e., if $\mathcal{I}$ is satisfiable, it must return “don't know" and (2) it returns “unsatisfiable" with high probability over the choice of the instance. As m increases, refutation becomes easier. The objective, then, is to refute instances with m as small as possible. This problem has been studied extensively and is related to hardness of approximation [17], proof complexity [8], statistical physics [13], cryptography [2], and learning theory [14]. As m increases, refutation becomes easier. The objective, then, is to refute instances with m as small as possible. Much research has focused on finding algorithms for refutation, especially in the special case of SAT; see [1] for references.

Most known refutation algorithms are based on semidefinite programming (SDP). For now, we think of an SDP relaxation of an instance $\mathcal{I}$ of $\mathrm{CSP}(P)$ as a black box that returns a number $\mathrm{SDPOpt} \in [0, 1]$ that approximates the maximum fraction of constraints that can be simultaneously satisfied. An SDP-based refutation algorithm takes a random instance $\mathcal{I}$ of $\mathrm{CSP}(P)$, solves some SDP relaxation of $\mathcal{I}$, and return “Unsatisfiable" if and only if $\mathrm{SDPOpt} < 1$. The majority of known polynomial-time algorithms for refutation fit into this framework (e.g., [1, 7, 20, 12, 18]). It is then natural to ask the following question.

> What is the minimum number of constraints needed by an efficient SDP-based refutation algorithm for $\mathrm{CSP}(P)$?

Allen et al. give an upper bound on the number of constraints required to refute an instance of $\mathrm{CSP}(P)$ in terms of a parameter $\mathrm{cmplx}(P)$ [1]. They define $\mathrm{cmplx}(P)$ to be the minimum t such that there is no t-wise uniform distribution over satisfying assignments to P. Clearly $1 \leq \mathrm{cmplx}(P) \leq k$ for nontrivial predicates and $\mathrm{cmplx}(P) = k$ when P is k-XOR or k-SAT. They give the following upper bound.

▶ **Theorem 1** ([1]). *There is an efficient SDP-based algorithm that refutes a random instance $\mathcal{I}$ of $\mathrm{CSP}(P)$ with high probability when $m \gg n^{\mathrm{cmplx}(P)/2}$.*

For special classes of predicates, we know that $n^{\mathrm{cmplx}(P)/2}$ constraints are also necessary for refutation by SDP-based algorithms. Schoenebeck considered arity-k predicates P whose satisfying assignments are a superset of k-XOR's; these include k-SAT and k-XOR. For such predicates, he showed that polynomial-size sum of squares (SOS) SDP relaxations cannot refute random instances with $m \leq n^{k/2-\varepsilon}$ [28] using a proof previously discovered by Grigoriev [21]. Based on work of Lee, Raghavendra, and Steurer [24], this implies that no polynomial-size SDP can be used to refute random instances of k-XOR or k-SAT when $m \leq n^{k/2-\varepsilon}$. This leads us to make the following conjecture.

▶ **Conjecture 2.** *Let ε be a constant greater than 0. Given a random instance $\mathcal{I}$ of $\mathrm{CSP}(P)$ with $m \leq n^{\mathrm{cmplx}(P)/2-\varepsilon}$, with high probability any polynomial-size SDP relaxation of $\mathcal{I}$ has optimal value 1 and can therefore not be used to refute $\mathcal{I}$.*

Proving this conjecture would essentially complete our understanding of the power of SDP-based refutation algorithms. To do this, it suffices to prove it for SOS SDP relaxations, as the SOS relaxation of $\mathrm{CSP}(P)$ is at least as powerful as an arbitrary SDP relaxation of comparable size [24]. Prior to this work, this SOS version Conjecture 2 appeared in [1]; we know of no other mention of this conjecture in the literature.[1]

Some partial progress has been made toward proving this conjecture. Building on results of Benabbas et al. [9] and Tulsiani and Worah [30], O'Donnell and Witmer proved lower bounds for the Sherali-Adams (SA) linear programming (LP) hierarchy and the Sherali-Adams$_+$ (SA$_+$) and Lovász-Schrijver$_+$ (LS$_+$) SDP hierarchies. All three of these hierarchies are weaker than SOS. The SA$_+$ hierarchy gives an optimal approximation for any CSP in the worst case assuming the Unique Games Conjecture [27]. They showed that Sherali-Adams linear programming (LP) relaxations cannot refute random instances with $m \leq n^{\mathrm{cmplx}(P)/2-\varepsilon}$ [26]; this implies that no polynomial-size LP can refute random instances with with $m \leq n^{\mathrm{cmplx}(P)/2-\varepsilon}$ by work of Chan et al. [11]. They showed that SA$_+$ cannot refute random instances with $m \leq n^{\mathrm{cmplx}(P)/2-\varepsilon}$ when a set of $o(m)$ constraints has been removed [26]. Also, they proved that SA$_+$ and LS$_+$ cannot refute fully random instances with $m \leq \Omega(n^{\mathrm{cmplx}(P)/2-1/3-\varepsilon})$. Much less is known for SOS. For predicates P that support a pairwise uniform distribution over satisfying assignments, Barak, Chan, and Kothari showed that polynomial-size SOS relaxations cannot refute random instances with $m = \Omega(n)$ in which $o(m)$ constraints have been removed [5].

In addition, there is a long history of related work on lower bounds for refutation in proof complexity (e.g., [21, 23, 10, 30]). Specifically, SA, SA$_+$, LS$_+$, and SOS have corresponding static semialgebraic proof systems and proving integrality gaps for these LP and SDP relaxations in equivalent to proving rank lower bounds for refutations in these proof systems.

Results

Our contribution is two-fold: First, we remove the assumption that a small number of constraints are deleted to show that fully-random CSP instances have integrality gaps in SA$_+$ for $m \leq \Omega(n^{t/2-\varepsilon})$.

▶ **Theorem 3.** *Let $P : [q]^k \to \{0,1\}$ be $(t-1)$-wise uniform-supporting and let $\mathcal{I}$ be a random instance of* CSP(P) *with n variables and $m \leq \Omega\left(n^{t/2-\varepsilon}\right)$ constraints. Then with high probability the* SA$_+$ *relaxation for $\mathcal{I}$ has value* 1*, even after $\Omega(n^{\frac{\varepsilon}{t-2}})$ rounds.*

Second, we use this result to show that fully random instances have LS$_+$ integrality gaps for $m \leq \Omega(n^{t/2-\varepsilon})$. Recall that LS$_+$ gives relaxations of 0/1-valued integer programs, so we restrict our attention here to Boolean CSPs with $P : \{0,1\}^k \to \{0,1\}$.

▶ **Theorem 4.** *Let $P : \{0,1\}^k \to \{0,1\}$ be $(t-1)$-wise uniform-supporting and let $\mathcal{I}$ be a random instance of* CSP(P) *with n variables and $m \leq \Omega\left(n^{t/2-\varepsilon}\right)$ constraints. Then with high probability the* LS$_+$ *relaxation for $\mathcal{I}$ has value* 1*, even after $\Omega(n^{\frac{\varepsilon}{t-2}})$ rounds.*

In their strongest form, our results hold for a static variant of the LS$_+$ SDP hierarchy that is at least as strong as both SA$_+$ and LS$_+$. We define this static LS$_+$ hierarchy in Section 2.

▶ **Theorem 5.** *Let $P : \{0,1\}^k \to \{0,1\}$ be $(t-1)$-wise uniform-supporting and let $\mathcal{I}$ be a random instance of* CSP(P) *with n variables and $m \leq \Omega\left(n^{t/2-\varepsilon}\right)$ constraints. Then with high probability the static* LS$_+$ *relaxation for $\mathcal{I}$ has value* 1*, even after $\Omega(n^{\frac{\varepsilon}{t-2}})$ rounds.*

[1] Barak, Kindler, and Steurer [6] made a related but different conjecture that the basic SDP relaxation is optimal for random CSPs.

Tulsiani and Worah proved this theorem in the special case of pairwise independence and $O(n)$ constraints [30, Theorem 3.27].[2]

These results provide further evidence for Conjecture 2 and, in particular, give the first examples of SDP hierarchies that are unable to refute CSPs with $(t-1)$-wise uniform-supporting predicates when $m \leq \Omega\left(n^{t/2-\varepsilon}\right)$.

From a dual point of view, we can think of SA_+, LS_+, and static LS_+ as semialgebraic proof systems and our results can be equivalently stated as rank lower bounds for these proof systems.

▶ **Theorem 6.** *Let $P : \{0,1\}^k \to \{0,1\}$ be $(t-1)$-wise uniform-supporting and let $\mathcal{I}$ be a random instance of $\mathrm{CSP}(P)$ with n variables and $m \leq \Omega\left(n^{t/2-\varepsilon}\right)$ constraints. Then with high probability any SA_+, LS_+, or static LS_+ refutation of $\mathcal{I}$ requires rank $\Omega(n^{\frac{\varepsilon}{t-2}})$.*

In another line of work, Feldman, Perkins, and Vempala [19] showed that if a predicate P is $(t-1)$-wise uniform supporting, then any statistical algorithm based on an oracle taking L values requires $m \geq \widetilde{O}\left(\frac{n^r}{L}\right)$ to refute. They further show that the dimension of any convex program refuting such a CSP must be at least $\widetilde{O}(n^{t/2})$. These lower bounds are incomparable to the integrality gap results stated above: While statistical algorithms and arbitrary convex relaxations are more general models, standard SDP hierarchy relaxations for k-CSPs, including the SA_+ and LS_+ relaxations we study, have dimension $\Theta(n^k)$ and are therefore not ruled out by this work.

Techniques

For simplicity we consider our CSP to have ± 1-values variables and $P : \{-1,1\}^k \to \{0,1\}$. To solve $\mathrm{CSP}(P)$ exactly, it suffices to optimize over distributions on assignments $\{0,1\}^n$. This, of course, is hard, so relaxations like SA, SA_+, LS_+, and SOS instead optimize over "pseudodistributions" on assignments [4], which are objects that look like distributions to simple enough functions. We can also define "pseudoexpectations" $\widetilde{\mathbb{E}}[\cdot]$ over these pseudodistributions. As the rank or degree of the relaxation increases, these pseudodistributions look more like actual distributions over $\{0,1\}^n$. However, the rank-r relaxations have size $n^{O(r)}$. The r-round SA_+ relaxation requires that $\widetilde{\mathbb{E}}[f(x)] \geq 0$ for $f(x) : \{0,1\}^n \to \mathbb{R}$ such that either (1) f depends on at most r variables or (2) f is the square of some affine function. We know that when $m \leq n^{\mathrm{cmplx}(P)/2-\varepsilon}$, there exist pseudodistributions satisfying (1) [9, 26]. Condition (2) is equivalent to positive semidefiniteness of the matrix of second pseudomoments $\widetilde{\mathbb{E}}[x_i x_j]$ of the pseudodistribution. Recalling that we are now considering ± 1-valued variables, previous work had constructed pseudodistributions and proved that their second moment matrix was positive semidefinite (PSD) by showing that it was diagonal and had nonnegative entries. The second moment matrix is diagonal when there are no correlations between assignments to pairs of variables under the pseudodistribution and the marginals of pseudodistribution for each variable is unbiased. This condition holds for instances with low densities, but correlations between variables arise as the density increases.

We show positive semidefiniteness in the presence of these correlations by showing that they must remain local. Our argument extends a technique of Tulsiani and Worah [30]. We prove that the graph induced by correlations between variables must have small connected components, each of which corresponds to a small block of off-diagonal nonzero

[2] Actually, [30] prove a rank lower bound for the dual static LS_+ proof system, but this is equivalent to a rank lower bound for the static LS_+ SDP hierarchy we consider here.

entries in the second pseudomoment matrix. Since each of these off-diagonal blocks is small, condition (1) guarantees that functions supported on these variables will have nonnegative pseudoexpectation. This means that each of these off-diagonal blocks is a PSD matrix. The pseudomoment matrix must then be PSD: It can be written as a sum of a diagonal matrix with nonnegative elements and PSD "correction matrices", each of which corresponds to a small connected component in the correlation graph.

2 Preliminaries

2.1 Constraint satisfaction problems

Given a predicate $P : [q]^k \to \{0,1\}$, an instance $\mathcal{I}$ of the CSP(P) problem with variables $x_1, \ldots, x_n$ is a multiset of P-constraints. Each P-constraint is a tuple (c, S), where $c \in [q]^k$ is the negation pattern and $S \in [n]^k$ is the scope. The corresponding constraint is $P(x_S + c) = 1$, where $x_S = (x_i)_{i \in S}$ and $+$ is component-wise addition mod q.

In the decision version of CSP(P), we wish to determine whether there exists an assignment satisfying all constraints of a given instance $\mathcal{I}$. In the optimization version, the objective is to maximize the fraction of simultaneously satisfied constraints. That is, we define $\mathrm{Val}_{\mathcal{I}}(x) := \frac{1}{m} \sum_{(c,S) \in \mathcal{I}} P(c + x_S)$ and wish to find $x \in [q]^n$ maximizing $\mathrm{Val}_{\mathcal{I}}(x)$. We will write $\max_x \mathrm{Val}_{\mathcal{I}}(x)$ as $\mathrm{Opt}(\mathcal{I})$.

Next, we define our random model. We consider instances in which m constraints are drawn independently and uniformly at random from among all $q^k n^k$ possible constraints with replacement. We distinguish between different orderings of the scope, as P may not be symmetric. The specific details of this definition are not important; our results hold for any similar model. For example, see [1, Appendix D]. A random instance is likely to be highly unsatisfiable: It is easy to show that $\mathrm{Opt}(\mathcal{I}) = \frac{|P^{-1}(1)|}{q^k} + o(1)$ for $m \geq O(n)$ with high probability.

Given an instance $\mathcal{I}$, we can consider the associated k-uniform hypergraph $H_{\mathcal{I}}$. This is the hypergraph on $V = [n]$ that has a hyperedge S if and only if S is the scope of some constraint of $\mathcal{I}$.

Next, we define the main condition on predicates that we will study.

▶ **Definition 7.** A predicate $P : [q]^k \to \{0,1\}$ is t-wise uniform supporting if there exists a distribution μ over $[q]^k$ supported on P's satisfying assignments such that $\Pr_{z \sim \mu}[z_T = \alpha] = q^{-|T|}$ for all $\alpha \in [q]^{|T|}$ and for all $T \subseteq [k]$ with $1 \leq |T| \leq t$.

2.2 LP and SDP hierarchies

We can define LP and SDP relaxations of both the decision and optimization versions of CSPs. All of our results will apply to both.

2.2.1 Representing CSP$(P)$with polynomial inequalities

All of the hierarchies we look at start with an initial set of polynomial inequalities representing an instance of a CSP and then iteratively tighten this relaxation. In this section, we will describe standard ways of constructing these initial relaxations.

To write down LP and SDP relaxations of CSP decision problems, we will need to represent the constraints of an instance $\mathcal{I}$ of CSP(P) as a set of polynomial inequalities. We will do this in two ways.

In SA, SA$_+$, and LS$_+$, we will represent each constraint as a degree-k polynomial inequality. Let $P'(x)$ be the unique degree-k polynomial such that $P'(z) = P(z)$ for all $z \in \{0,1\}^k$. Also, given $a \in [0,1]^k$ and $b \in \{0,1\}$, use $a^{(b)}$ to denote a if $b = 0$ and $1 - a$ if $b = 1$. For $z \in [0,1]^k$ and $c \in \{0,1\}^k$, let $z^{(c)} \in [0,1]^k$ be such that $(z^{(c)})_i = z_i^{(c_i)}$. Then we define the base degree-k relaxation of $\mathcal{I}$ to be the set

$$R_{\mathcal{I}} := \{x \in [0,1]^n \mid P'(x_S^{(c)}) = 1 \ \forall (c,S) \in \mathcal{I}\}. \tag{2.1}$$

It will be most natural to construct SA, SA$_+$, and static LS$_+$ relaxations starting from this polytope. In addition, this formulation immediately generalizes to larger alphabets.

For LS$_+$, on the other hand, we will have to start with a *linear* relaxation. Recall that any nontrivial arity-k predicate P can be represented as a conjunction of at most $2^k - 1$ disjunctions of arity k. In particular, letting $F = \{z \in \{0,1\}^k \mid P(z) = 0\}$, we see that

$$P(z) = \bigwedge_{f \in F} \bigvee_{i=1}^{k} f_i \oplus z_i. \tag{2.2}$$

Using (2.2), we can represent $\mathcal{I}$ as a k-SAT instance with at most $(2^k - 1) \cdot m$ constraints. The linear relaxation we will consider is the standard linear relaxation for this k-SAT instance. For each clause $\bigvee_{i=1}^{k} c_i \oplus z_i$, we will add the inequality $\sum_{i=1}^{k} z_i^{(c_i)} \geq 1$. We then obtain the following linear relaxation for $\mathcal{I}$.

$$L_{\mathcal{I}} := \left\{ x \in [0,1]^n \;\middle|\; \sum_{i=1}^{k} x_{S_i}^{(c_i \oplus f_i)} \geq 1 \ \forall (c,S) \in \mathcal{I}, f \in F \right\}. \tag{2.3}$$

To get a maximization version of the LS$_+$ relaxation, we will need a base relaxation with linear constraints and a linear objective function. To do this, we will start with $L_{\mathcal{I}}$ and add variables $z_{c,S}$ for all constraints $(c,S) \in \mathcal{I}$ and consider the following polytope in $\mathbb{R}^{n+m}$.

$$L'_{\mathcal{I}} := \left\{ (x,z) \in [0,1]^{n+m} \;\middle|\; \sum_{i=1}^{k} x_{S_i}^{(c_i \oplus f_i)} \geq z_{c,S} \ \forall (c,S) \in \mathcal{I}, f \in F \right\}. \tag{2.4}$$

We can then write the following standard LP relaxation of CSP(P).

$$\begin{aligned} &\max \frac{1}{m} \sum_{(c,S) \in \mathcal{I}} z_{c,S} \\ &\text{s.t. } (x,z) \in L'_{\mathcal{I}} \end{aligned}$$

2.2.2 Sherali-Adams

The Sherali-Adams (SA) linear programming hierarchy gives a family of locally consistent distributions on assignments to sets of variables. As the size of these sets increases, the relaxation becomes tighter.

▶ **Definition 8.** Let $\{D_S\}$ be a family of distributions D_S over $[q]^S$ for all $S \subseteq [n]$ with $|S| \leq r$. We say that $\{D_S\}$ is r-locally consistent if for all $T \subseteq S \subseteq [n]$ with $|S| \leq r$, the marginal of D_S on T is equal to D_T. That is,

$$D_T(\alpha) = \sum_{\substack{\beta \in [q]^S \\ \beta_T = \alpha}} D_S(\beta).$$

Given polynomial inequalities $a_1(x) \geq 0$, $a_2(x) \geq 0$, $\ldots$, $a_m(x) \geq 0$ called axioms such that each a_j depends on at most r variables, we consider the set

$$A := \{x \in \mathbb{R}^n \mid a_1(x) \geq 0, a_2(x) \geq 0, \ldots, a_m(x) \geq 0\}.$$

For a polynomial q, let $\operatorname{supp}(q)$ be the set of all variables on which a depends. We then define the *rank-r* SA *relaxation for A* as the set of all families of distributions $\{D_S\}_{S\subseteq[n],\,|S|\leq r}$ over $[q]^S$ satisfying following two properties.

1. $\{D_S\}_{S\subseteq[n],\,|S|\leq r}$ is r-locally consistent.
2. $\mathbb{E}_{\alpha\sim D_{\operatorname{supp}(a_j)}}[a_j(\alpha)] = 1$ for all $j \in [m]$.

We denote this set of families of distributions as $\mathrm{SA}^r(A)$; note that $\mathrm{SA}^r(A)$ is a polytope.

In the case of an instance $\mathcal{I}$ of $\mathrm{CSP}(P)$, we can write r-round SA relaxations in both feasibility and optimization forms. In the feasibility formulation, we check whether or not the polytope $\mathrm{SA}^r(R_\mathcal{I})$ is feasible; this is a relaxation of the problem of checking whether or not all constraints can be simultaneously satisfied.

In the rank-r SA optimization formulation, we solve the following LP.

$$\begin{aligned} \max \quad & \frac{1}{m}\sum_{(c,S)\in\mathcal{I}} \mathbb{E}_{\alpha\sim D_S}[P(\alpha+c)] \\ \text{s.t.} \quad & \{D_S\}_{S\subseteq[n],\,|S|\leq r} \in \mathrm{SA}^r(\emptyset). \end{aligned} \tag{2.5}$$

This is an LP of size $n^{O(r)}$ that is a relaxation of the problem of maximizing the number of satisfied constraints. As r increases, the number of variables and constraints in this LP increases and the relaxation tightens until $r = n$, when SA has $\Theta(q^n)$ variables and gives the exact solution.

2.2.3 Sherali-Adams$_+$

The Sherali-Adams$_+$ (SA$_+$) SDP hierarchy additionally requires the second moment matrix of these distributions to be PSD. Given a family of local distributions $\{D_S\}$, define $M(D) \in \mathbb{R}^{(nq+1)\times(nq+1)}$ to be the symmetric matrix indexed by $(0, [n] \times [q])$ such that

$$\begin{aligned} M(D)_{0,0} &= 1 \\ M(D)_{0,(i,a)} &= D_{\{i\}}(x_i = a) \\ M(D)_{(i,a),(j,b)} &= D_{\{i,j\}}(x_i = a \wedge x_j = b). \end{aligned}$$

Note that the $((i,a),(i,b))$-element of B is $D_{\{i\}}(x_i = a)$ if $a = b$ and is 0 if $a \neq b$. Given an initial set A of axioms as above, we define $\mathrm{SA}_+^r(A)$ as we did $\mathrm{SA}^r(A)$ with the following additional condition.

3. $M(D)$ is PSD.

We define the optimization version of the rank-r SA$_+$ relaxation analogously to the optimization version of SA in (2.5); this is an SDP with size $n^{O(r)}$.

We can equivalently define SA$_+$ by requiring the covariance matrix of the locally consistent $\{D_S\}$ distributions to be positive semidefinite (PSD).

▶ **Definition 9.** The covariance matrix Σ for r-locally consistent distributions $\{D_S\}$ for $r \geq 2$ is defined as

$$\Sigma_{(i,a),(j,b)} = D_{\{i,j\}}(x_i = a \wedge x_j = b) - D_{\{i\}}(x_i = a) \cdot D_{\{j\}}(x_j = b).$$

These two representations are equivalent [31].

▶ **Lemma 10.** *M is PSD if and only if Σ is PSD.*

We include the proof in Appendix B.

The covariance matrix condition will be more convenient for us to work with. A valid global distribution has a PSD covariance matrix, so SA_+ is a relaxation of $\mathrm{CSP}(P)$ and is exact for $r = n$.

2.2.4 Lovász-Schrijver$_+$

We now define the Lovász-Schrijver$_+$ (LS_+) SDP relaxation for binary CSPs whose variables are 0/1-valued. Given an initial polytope $K \in \mathbb{R}^n$, we would like to generate a sequence of progressively tighter relaxations. To define one LS_+ lift-and-project step, we will use the cone

$$\widetilde{K} = \{(\lambda, \lambda x_1, \dots, \lambda x_n) \mid \lambda > 0, x_1, \dots, x_n \in K\}.$$

K can be recovered by taking the intersection with $x_0 = 1$.

▶ **Definition 11.** Let $\widetilde{K}$ be a convex cone in $\mathbb{R}^{n+1}$. Then the lifted LS_+ cone $N_+(\widetilde{K})$ is the cone of all $y \in \mathbb{R}^{n+1}$ for which there exists an $(n+1) \times (n+1)$ matrix Y satisfying the following:

1. Y is symmetric and positive semidefinite.
2. For all i, $Y_{ii} = Y_{i0} = y_i$.
3. For all i, $Y_i \in \widetilde{K}$ and $Y_0 - Y_i \in \widetilde{K}$

where Y_i is the ith column of Y. Then we define $N_+(K)$ to be $N_+(\widetilde{K}) \cap \{x_0 = 1\}$. The r-round LS_+ relaxation of a polytope K results from applying the N_+ operator r times. That is, we define $N_+^r(K) = N_+(N_+^{r-1}(K))$. Y is called a protection matrix for y.

A solution to the r-round LS_+ relaxation for a polytope $K \in \mathbb{R}^n$ defined by $n^{O(1)}$ linear constraints can be computed in time $n^{O(r)}$ using an SDP.

We can write an LS_+ relaxation for $\mathrm{CSP}(P)$ in two ways. The maximization version of the LS_+ CSP relaxation is

$$\begin{aligned} \max\ & \frac{1}{m} \sum_{(c,S) \in \mathcal{I}} z_{c,S} \\ \text{s.t. } & (x, z) \in N_+^r(L'_{\mathcal{I}}). \end{aligned}$$

Alternatively, we can check feasibility of $N_+^r(L_{\mathcal{I}})$.

We note here that though it is more natural to apply SA, SA_+, and static LS_+ to (2.1), applying SA, SA_+, and static LS_+ to (2.3) yields a relaxation that is at least as strong.

▶ **Lemma 12.** *Let $r \geq k$. Then the following statements hold.*

1. $\mathrm{SA}^r(R_{\mathcal{I}}) = \mathrm{SA}^r(L_{\mathcal{I}})$.
2. $\mathrm{SA}_+^r(R_{\mathcal{I}}) = \mathrm{SA}_+^r(L_{\mathcal{I}})$.
3. $\mathrm{StaticLS}_+^r(R_{\mathcal{I}}) = \mathrm{StaticLS}_+^r(L_{\mathcal{I}})$

We include the proof in Appendix D. The $\mathrm{StaticLS}_+^r$ operator is defined in the next section.

2.2.5 Static LS$_+$

The static LS$_+$ relaxation strengthens both SA$_+$ and LS$_+$. As in the case of SA$_+$, we start with a family of r-locally consistent distributions and then further require that they satisfy certain positive semidefiniteness constraints. In particular, for all $X \subseteq [n]$ with $|X| \leq r-2$ and all $\alpha \in [q]^X$, define the matrices $M_{X,\alpha}(D) \in \mathbb{R}^{(nq+1)\times(nq+1)}$ as follows.

$$\begin{aligned} M_{X,\alpha}(D)_{0,0} &= 1 \\ M_{X,\alpha}(D)_{0,(i,a)} &= D_{\{i\}\cup X}(x_i = a \wedge X = \alpha) \\ M_{X,\alpha}(D)_{(i,a),(j,b)} &= D_{\{i,j\}\cup X}(x_i = a \wedge x_j = b \wedge X = \alpha). \end{aligned}$$

In addition to the SA constraints, the r-round static LS$_+$ relaxation StaticLS$^r_+(A)$ satisfies the following constraint.

3'. $M_{X,\alpha}(D)$ is PSD for all $X \subseteq [n]$ with $|X| \leq r-2$ and all $\alpha \in [q]^X$.

Observe that these positive semidefiniteness constraints can be formulated as a positive semidefiniteness constraint for a single matrix. In particular, let M_{total} be the block diagonal matrix with the $M_{X,\alpha}$'s on the diagonal. Then M_{total} has size at most $(qn)^{O(r)}$ and M_{total} is PSD if and only if all of the $M_{X,\alpha}$'s are PSD. The maximization version is again defined exactly as it was for SA and SA$_+$. Unlike LS$_+$, this hierarchy immediately generalizes to non-binary alphabets.

For intuition, one can think of this hierarchy as requiring positive semidefiniteness of the covariance matrices of the conditional distributions formed by conditioning on the events that X is assigned α for all X with $|X| \leq r-2$ and all $\alpha \in [q]^X$. We prefer the definition presented here because it more easily handles the case of X getting assigned α with probability 0 in which the corresponding conditional distribution is not defined.

We note that we have not seen this hierarchy defined in this form in previous work, but it is dual to the static LS$_+$ proof system defined in [22] and described below in Section 2.3 (see Appendix E).

2.2.6 Pseudodistributions: An alternate point of view

We can equivalently define SA, SA$_+$, and static LS$_+$ in terms of pseudodistributions [4, 3]. A pseudodistribution is a map $\sigma : \{0,1\}^n \to \mathbb{R}$ that "looks like" a valid distribution over $\{0,1\}^n$ to simple enough functions. Define the corresponding pseudoexpectation $\widetilde{\mathbb{E}}_{\mathbf{x}\sim\sigma}[\cdot]$ as $\widetilde{\mathbb{E}}_{\mathbf{x}\sim\sigma}[f(\mathbf{x})] = \sum_{x\in\{0,1\}^n} \sigma(x)f(x)$.

Sherali-Adams

A *rank-r* SA *pseudodistribution* satisfies that following two conditions.

1. $\sum_{x\in\{0,1\}^n} \sigma(x) = 1$.
2. $\widetilde{\mathbb{E}}_{\mathbf{x}\sim\sigma}[f(\mathbf{x})] \geq 0$ for all nonnegative functions $f : \{0,1\}^n \to \mathbb{R}$ that depend on at most r variables.

Sherali-Adams$_+$

A *rank-r* SA$_+$ *pseudodistribution* satisfies Condition 1 and a stronger version of Condition 2:

2'. $\widetilde{\mathbb{E}}_{\mathbf{x}\sim\sigma}[f(\mathbf{x})] \geq 0$ for all nonnegative functions $f : \{0,1\}^n \to \mathbb{R}$ satisfying one of the following.
  1. f depends on at most r variables.
  2. $f = \ell^2$ for some function ℓ with degree at most 1.

Static LS$_+$

A *rank-r static* LS$_+$ *pseudodistribution* satisfies Condition 1 and a version of Condition 2 that is stronger still:

2″. $\widetilde{\mathbb{E}}_{\mathbf{x}\sim\sigma}[f(\mathbf{x})] \geq 0$ for all nonnegative functions $f : \{0,1\}^n \to \mathbb{R}$ satisfying one of the following.
  1. f depends on at most r variables.
  2. There exists a set X of $r-2$ variables such that for any assignment α to X, the function $f_{X,\alpha} : [q]^{[n]\setminus X} \to \mathbb{R}$ resulting from setting X to α is equal to ℓ^2 for some function ℓ with degree at most 1 possibly depending on X and α.

All three relaxations maximize the objective function

$$\frac{1}{m}\sum_{(c,S)\in\mathcal{I}} \widetilde{\mathbb{E}}_{\mathbf{x}\sim\sigma}[P(x_S+c)]$$

over their corresponding pseudodistributions σ.

2.3 The dual point of view: Static semialgebraic proof systems

We consider refutation of CSPs via semialgebraic proof systems. Starting from a set of axioms $\{a_i(x)\geq 0\}$ that capture the constraints of the CSP as polynomial inequalities, semialgebraic proof systems derive new inequalities the are implied by the axioms and integrality of the variables. To prove that an instance is unsatisfiable, we wish to derive the contradiction $-1\geq 0$. We consider the SA, SA$_+$, LS$_+$, and static LS$_+$ proof systems. Here, we will only deal with $\{0,1\}$-valued variables.

The SA proof system

A SA refutation has the form

$$\sum_{\ell}\gamma_\ell a_\ell(x)\phi_{I_\ell,J_\ell}(x) = -1,$$

where $\gamma_\ell \geq 0$, a_ℓ is an axiom, and $\phi_{I_\ell,J_\ell} = \prod_{i\in I_\ell} x_i \prod_{j\in J_\ell}(1-x_j)$. This is a proof of unsatisfiability because under the assumption that the all x_i variables are in $\{0,1\}$, every term of the above sum most be nonnegative and it is therefore a contradiction. The rank of this proof (often called the degree) is the maximum degree of any of the terms. The *size* of the proof is the number of terms in the sum; this follows from Farkas' Lemma. Static SA proofs is automatizable: A rank r SA proof may be found in time $n^{O(r)}$ if it exists by solving an LP. The SA proof system first appeared in [22] with the name static LS$^\infty$; the dual hierarchy of LP relaxations was introduced by [29]. A rank-r SA refutation exists if and only if the corresponding rank-r SA relaxation is infeasible.

The SA$_+$ proof system

In SA$_+$, a proof has the form

$$\sum_{\ell}\gamma_\ell a_\ell(x)\phi_{I_\ell,J_\ell}(x) + \sum_{s}\nu_s\lambda_s(x)^2 = -1,$$

where $\gamma_\ell,\nu_s\geq 0$ and the λ_s's are affine functions. The rank of the proof is its degree. The dual SA$_+$ hierarchy of SDP relaxation first appeared in [27]. Again, a rank-r SA$_+$ refutation exists if and only if the corresponding rank-r SA$_+$ relaxation is infeasible. SA$_+$ proofs of rank r can be found in time $n^{O(r)}$ if they exist.

The LS_+ proof system

The LS_+ proof system [25] is dynamic, meaning that a proof is built up over a series of steps. A proof in LS_+ is a sequence of expressions of the form $P(x) \geq 0$. A new inequality is derived from the inequalities already in the proof using inference rules. When $\deg(P(x)) \leq 1$, we allow the following:

$$\frac{P(x) \geq 0}{x_i \cdot P(x) \geq 0} \qquad \frac{P(x) \geq 0}{(1-x_i) \cdot P(x) \geq 0} \qquad \frac{}{P(x)^2 \geq 0.}$$

We also allow nonnegative linear combinations:

$$\frac{P(x) \geq 0 \quad Q(x) \geq 0}{\alpha \cdot P(x) + \beta \cdot Q(x) \geq 0}$$

for $\alpha, \beta \geq 0$. An LS_+ proof is therefore a sequence of “lifting" steps in which we multiply by some x_i or $(1 - x_i)$ to get a degree-2 polynomial and “projection" steps in which we take nonnegative linear combinations to reduce the degree back to 1. We can view an LS_+ proof as a DAG with inequalities at each vertex and $-1 \geq 0$ at the root. The rank of an LS_+ proof is the maximum number of lifting steps in any path to the root. The LS_+ proof system is not known to be automatizable; see Section 8 of [10] for details. An rank-r LS_+ refutation exists if and only if the corresponding rank-r LS_+ relaxation is infeasible [15].

The static LS_+ proof system

A static LS_+ proof [22] has the following form.

$$\sum_{\ell} \gamma_\ell b_\ell(x) \phi_{I_\ell, J_\ell}(x) = -1,$$

where $\gamma_\ell \geq 0$, b_ℓ is an axiom or the square of an affine function, and ϕ_{I_ℓ,J_ℓ} is as above. Note that this proof system as at least as powerful as the SA_+ proof system: Terms in the sum may be products of a $\phi_{I,J}$ term and the square of an affine function instead of just the square of an affine function or just an axiom multiplied by a $\phi_{I,J}$ term. We do not know of any results on the automatizability of static LS_+. Once again, there exists a static LS_+ refutation if and only if the corresponding static LS_+ relaxation is infeasible. We do not know of any proof of this statement in the literature, so we include one in Appendix E.

2.4 Expansion

Given a set of constraints T, we define its neighbor set $\Gamma(T)$ as $\Gamma(T) := \{v \in [n] \mid v \in \mathrm{supp}(C) \text{ for some } C \in T\}$. We can then define expansion.

▶ **Definition 13.** An instance $\mathcal{I}$ of $\mathrm{CSP}(P)$ is (s, e)-expanding if for every set of constraints T with $|T| \leq s$, $|\Gamma(T)| \geq e|T|$.

We can also define T's boundary neighbors as $\partial T := \{v \in [n] \mid v \in \mathrm{supp}(C) \text{ for exactly one } C \in T\}$ and give a corresponding notion of boundary expansion.

▶ **Definition 14.** An instance $\mathcal{I}$ of $\mathrm{CSP}(P)$ is (s, e)-boundary expanding if for every set of constraints T with $|T| \leq s$, $|\partial T| \geq e|T|$.

We state a well-known connection between expansion and boundary expansion stated in, e.g., [30].

▶ **Fact 15.** *$(s, k-d)$-expansion implies $(s, k-2d)$-boundary expansion.*

It is also well-known that randomly-chosen sets of constraints have high expansion [9, 26]:

▶ **Lemma 16.** *Fix $\delta > 0$. With high probability, a set of $m \leq \Omega(n^{t/2-\delta})$ constraints chosen uniformly at random is both $\left(n^{\frac{\delta}{t-2}}, k - \frac{t}{2} + \frac{\delta}{2}\right)$-expanding and $\left(n^{\frac{\delta}{t-2}}, k - t + \delta\right)$-boundary expanding.*

We give proofs of both of these statements in Appendix A.

2.5 Constructing consistent local distributions

Here, we recall a construction of consistent local distributions supported on satisfying assignments. This construction was first given in [9] and has been used in many subsequent works (e.g, [30, 26, 5]). In Appendix C, we give proofs of all results mentioned in this section.

We first need to define the notion of a closure of a set of variables. For $S \subseteq [n]$, let $H_{\mathcal{I}} - S$ denote the hypergraph $H_{\mathcal{I}}$ with the vertices of S and all hyperedges contained in S removed. Intuitively, the closure of a set $S \subseteq [n]$ is a superset of S that is not too much larger than S isn't very well-connected to the rest of the instance in the sense that $H_{\mathcal{I}} - S$ has high expansion.

▶ **Lemma 17** ([9, 30]). *If $H_{\mathcal{I}}$ is (s_1, e_1)-expanding and S is a set of variables such that $|S| < (e_1 - e_2)s_1$ for some $e_2 \in (0, e_1)$, then there exists a set $\mathsf{Cl}(S) \subseteq [n]$ such that $S \subseteq \mathsf{Cl}(S)$ and $H_{\mathcal{I}} - \mathsf{Cl}(S)$ is (s_2, e_2)-expanding with $s_2 \geq s_1 - \frac{|S|}{e_1 - e_2}$ and $\mathsf{Cl}(S) \leq \frac{k+2e_1-e_2}{2(e_1-e_2)}|S|$.*

We now use the closure to define consistent local distributions supporting on satisfying assignments. We assume that there exists a $(t-1)$-wise independent distribution μ over satisfying assignments to P. For a constraint $C = (c, S)$, let μ_C be the distribution defined by $\mu_C(z) = \mu(z_1 + c_1, \ldots, z_k + c_k)$ and let $\mathcal{C}(S)$ be the set of constraints whose support is entirely contained within S. For a set of variables $S \subseteq [n]$ and an assignment $\alpha \in [q]^S$, we use the notation $S = \alpha$ to indicate the the variables of S are labeled according to the assignment α. For a constraint $C = (c, S)$ and an assignment α to a superset of S, let $\mu_C(\alpha) = \mu_C(\alpha_S)$.

For $S \subseteq [n]$, we can then define the distribution D'_S over $[q]^S$ as

$$D'_S(S = \alpha) = \frac{1}{Z_S} \sum_{\substack{\beta \in [q]^S \\ \beta_S = \alpha}} \prod_{C \in \mathcal{C}(S)} \mu_C(\beta), \text{ where } Z_S = \sum_{\beta \in [q]^S} \prod_{C \in \mathcal{C}(S)} \mu_C(\beta).$$

Using D', we can then define consistent local distributions D_S for $|S| \leq r$ so that $D_S(S = \alpha) = D'_{\mathsf{Cl}(S)}(S = \alpha)$. [9, 26] proved that these distributions are r-locally consistent for $r = n^{\frac{\varepsilon}{t-2}}$.

▶ **Theorem 18.** *For a random instance $\mathcal{I}$ with $m \leq \Omega(n^{t/2-\varepsilon})$, the family of distributions $\{D_S\}_{|S| \leq r}$ is r-locally consistent for $r = n^{\frac{\varepsilon}{t-2}}$ and is supported on satisfying assignments.*

This theorem shows that the SA cannot efficiently refute random $(t-1)$-wise uniforming supporting instances: the r-round SA LP still has value 1 for some $r = \Omega(n^{\frac{\varepsilon}{t-2}})$ when $m \leq \Omega(n^{t/2-\varepsilon})$. In this paper, we show that even when we add the SA$_+$ requirement that the covariance matrix is PSD, we still cannot refute when $m \leq \Omega(n^{t/2-\varepsilon})$.

As in [30], we can also construct r-locally consistent *conditional* distributions. We will only need these distributions in the proof of our LS$_+$ result, so we describe them only in the binary alphabet case. Let $S \subseteq [n]$, let $X \subseteq [n]$ be a subset of the variables such that $X \cap S = \emptyset$, and let $\alpha \in \{0,1\}^X$ be an assignment to X such that $\mu_C(\alpha) > 0$ for all constraints in $\mathcal{C}(X)$. Define $D_{S|X=\alpha}$ to be the distribution D_S conditioned on the event that X is assigned α under the distribution D_X. That is, $D_{S|X=\alpha}(S = \beta) = \frac{D_{S \cup X}(S=\beta \wedge X=\alpha)}{D_{S \cup X}(X=\alpha)}$.

▶ **Lemma 19** ([30, Lemma 3.13])**.** *Let $X \subseteq [n]$ and let $\{D_S\}$ be a family of r-locally consistent distributions for sets $S \subseteq [n]$ such that $S \cap X = \emptyset$ and $|S \cup X| \leq r$. Then the family of conditional distributions $\{D_{S|X=\alpha}\}$ is $(r - |X|)$-locally consistent for any $\alpha \in \{0,1\}^X$ such that $\mu_C(\alpha) > 0$ for all constraints C in $\mathcal{C}(X)$.*

The following is a simple corollary that we will use in Section 7:

▶ **Corollary 20.** *Let $\mathcal{I}$ be a random instance of* CSP(P) *with n variables and let $X \subseteq [n]$ such that $|X| \leq \Omega(n^{\frac{\varepsilon}{t-2}})$ and let $\alpha \in \{0,1\}^X$ be any assignment to X such that $\mu_C(\alpha) > 0$ for all constraints in $C \in \mathcal{C}(X)$. Then the family of conditional distributions $\{D_{S|X=\alpha}\}_{|S| \leq r, S \cap X = \emptyset}$ is r-locally consistent for some $r = \Omega(n^{\frac{\varepsilon}{t-2}})$.*

3 Overview of the proof

Previous work [9, 30] only considers instances with a linear number of constraints and relies on the fact that most pairs of variables are uncorrelated in this regime. For $m \gg n$, however, correlations between pairs of vertices do arise because the underlying hypergraph becomes more dense. The major technical contribution of this work is to deal with these correlations by proving that they remain local. More precisely, we consider the graph induced by correlations between variables: Two variables are connected if they have non-zero correlation. We prove that this graph must have connected components of at most constant size. Each of these connected components can then be covered by a local distribution of constant size and this suffices to ensure PSDness of the covariance matrix.

Showing that a set of local distributions is a valid SA$_+$ solution requires proving that these distributions are consistent and proving that their covariance matrix is PSD. Local consistency was proven in previous work [9, 26]. To prove Theorem 3, it remains to argue that the covariance matrix is PSD. The proof of this statement has three steps: First, we show in Section 4 that if the correlation graph has small connected components, then the covariance matrix is PSD. Second, we show any non-zero correlation must have been caused by a relatively dense subset of constraints in Section 5. In Section 6, we show that connected components in the correlation graph must be small or they would induce large dense subsets of constraints that would violate expansion properties.

In Section 7, we show that positive semidefiniteness of the covariance matrix implies Theorem 4.

4 The correlation graph

In this section, we define the correlation graph, and show that if the correlation graph only has small connected components, then the covariance matrix is PSD.

▶ **Definition 21.** The correlation graph G_c associated with r-locally consistent distributions $\{p_S\}$ is the graph on $[n]$ with an edge between every pair of variables for which there is a non-zero entry in the covariance matrix for $\{p_S\}$. More formally,

$$E(G_c) = \{(u,v) \in [n] \times [n] \mid u \neq v, \exists (a,b) \in [q] \times [q] \text{ s.t. } \Sigma_{(u,a),(v,b)} \neq 0\}.$$

▶ **Lemma 22.** *Let $\{p_S\}$ be a family of r-locally consistent distributions. If all connected components in the correlation graph associated with $\{p_S\}$ have size at most r, then the covariance matrix for $\{p_S\}$ is PSD.*

Proof. Consider the partition $V_1, V_2, \ldots, V_\ell$ of $[n]$ such that u and v are in the same set if and only if they are connected in the correlation graph. We then have nonzero entries in the covariance matrix only for pairs $((u,a),(v,b))$ such that $u, v \in V_i$ for some i. Ordering the rows and columns of the covariance matrix according to the partition, we see that the covariance matrix is block diagonal with a nonzero block on the diagonal for each connected component of the correlation graph. Each of these blocks is PSD since each is the covariance matrix of the Sherali-Adams distribution p_{V_i} for the corresponding set V_i of the partition with size at most r and the covariance matrix of valid distribution is always PSD. Since each block is PSD, the entire matrix is PSD. ◀

We already know that $\{D_S\}$ defined in Section 2.5 is r-locally consistent with high probability when $m \le \Omega(n^{t/2-\epsilon})$. In the following sections, we will show that connected components in the correlation graph associated with $\{D_S\}$ is small. Hence, from Lemma 22, $\{D_S\}$ is feasible solution for SA$_+$ SDP while it gives the trivial objective value 1.

5 Correlations are induced by small, dense structures

In this section, we show that pairwise correlations in $\{D_S\}$ are only generated by small, dense subhypergraphs that we will call "bad structures". Given a set of hyperedges W, call a variable v an W-boundary variable if it is contained in exactly one constraint in W.

▶ **Definition 23.** For variables u and v, a bad structure for u and v is a set of constraints W satisfying the following properties:

1. $u, v \in \Gamma(W)$.
2. The hypergraph induced by W is connected.
3. Every constraint contains at most $k-t$ W-boundary variables except for u and v.

We also say W is a bad structure if W is a bad structure for some u and v.

A bad structure for u and v generates correlation between u and v with respect to $\{D_S\}$.

▶ **Lemma 24.** *If there is no bad structure for u and v of size at most $|\mathcal{C}(\mathsf{Cl}(\{u,v\}))|$, then u and v are not correlated with respect to $D_{\{u,v\}}$.*

We need the following technical claim, which states that the distribution D'_S isn't affected removing a constraint with many boundary variables.

▶ **Claim 25.** *Let $T \subseteq S \subseteq [n]$ be sets of variables. Let $C^* \in \mathcal{C}(S)$ be some constraint covered by S. If $|(\partial\mathcal{C}(S) \cap C^*) \setminus T| \ge k-t+1$, then for any $\alpha \in \{0,1\}^T$,*

$$D'_S(\alpha) = \frac{1}{q^{|T\cap(\partial\mathcal{C}(S)\cap C^*)|}} \cdot D'_{S\setminus(\partial\mathcal{C}(S)\cap C^*)}(\alpha_{T\setminus(\partial\mathcal{C}(S)\cap C^*)}),$$

Proof. Let $B = \partial\mathcal{C}(S) \cap C^*$ be the boundary variables of $\mathcal{C}(S)$ contributed by C^*, i.e., the variables contained in C^* that don't appear in any other constraint of $\mathcal{C}(S)$. First, note that

$$\begin{aligned}\sum_{\substack{\beta\in\{0,1\}^S\\ \beta_T=\alpha}} \prod_{C\in\mathcal{C}(S)} \mu_C(\beta) &= \sum_{\substack{\beta\in\{0,1\}^{S\setminus B}\\ \beta_{T\setminus B}=\alpha_{T\setminus B}}} \prod_{c\in\mathcal{C}(S)\setminus\{C^*\}} \mu_C(\beta) \sum_{\substack{\gamma\in\{0,1\}^B\\ \gamma_{B\cap T}=\alpha_{B\cap T}}} \mu_{C^*}(\beta,\gamma)\\ &= \frac{1}{q^{k-|B\setminus T|}} \sum_{\substack{\beta\in\{0,1\}^{S\setminus B}\\ \beta_{T\setminus B}=\alpha_{T\setminus B}}} \prod_{C\in\mathcal{C}(S)\setminus\{C^*\}} \mu_C(\beta). \end{aligned} \quad (5.6)$$

The last line holds because $|B \setminus T| \geq k - t + 1$ and μ is $(t-1)$-wise independent.

Similarly,

$$Z_S = \sum_{\beta \in \{0,1\}^S} \prod_{C \in \mathcal{C}(S)} \mu_C(\beta) = \frac{1}{q^{k-|B|}} \sum_{\beta \in \{0,1\}^{S \setminus B}} \prod_{C \in \mathcal{C}(S) \setminus \{C^*\}} \mu_C(\beta). \tag{5.7}$$

Dividing (5.6) by (5.7), we see that $D'_S(\alpha) = \frac{1}{q^{|B \cap T|}} \cdot D'_{S \setminus B}(\alpha_{T \setminus B})$. ◀

Using Claim 25, we prove Lemma 24.

Proof of Lemma 24. Let $S_0 = \mathcal{C}(\mathsf{Cl}(\{u,v\}))$. Say there exists a constraint C_1 such that $|(\partial \mathcal{C}(S_0) \cap C_1) \setminus \{u,v\}| \geq k - t + 1$. Let $S_1 = S_0 \setminus (\partial \mathcal{C}(S_0) \cap C_1)$. If there exists a constraint C_2 such that $|(\partial \mathcal{C}(S_1) \cap C_2) \setminus \{u,v\}| \geq k - t + 1$, remove its boundary variables in the same manner to get S_2. Continue in this way until we obtain a set S_ℓ such that $|(\partial \mathcal{C}(S_\ell) \cap C) \setminus \{u,v\}| \leq k - t$ for every constraint $C \in \mathcal{C}(S_\ell)$ ($\mathcal{C}(S_\ell)$ could be empty). Since $|(\partial \mathcal{C}(S_{i-1}) \cap C_i) \setminus \{u,v\}| \geq k - t + 1$ for $1 \leq i \leq \ell$, we can apply Claim 25 ℓ times to see that

$$D_{\{u,v\}}(u = a \wedge v = b) = \begin{cases} \frac{1}{q} \cdot D'_{S_\ell}(u = a) & \text{if } u \in S_\ell, v \notin S_\ell \\ \frac{1}{q} \cdot D'_{S_\ell}(v = b) & \text{if } v \in S_\ell, u \notin S_\ell \\ \frac{1}{q^2} & \text{if } u, v \notin S_\ell \\ D'_{S_\ell}(u = a \wedge v = b) & \text{if } u, v \in S_\ell. \end{cases}$$

In the first three cases, it is easy to see that the lemma holds. In the last case, since $\mathcal{C}(S_\ell)$ cannot be a bad structure, the hypergraph induced by S_ℓ must be disconnected. Say $U_1, U_2, \ldots, U_t$ are the vertex sets of the connected components of S_ℓ. Remove all connected components that contain neither u nor v to get S'_ℓ. Again, the hypergraph induced by S'_ℓ cannot be connected; otherwise, $\mathcal{C}(S'_\ell)$ would be a bad structure. This means that S'_ℓ must be disconnected with u and v in different components. Say S'_u and S'_v are the vertex sets of the connected components of S_ℓ containing u and v, respectively. Then

$$D_{\{u,v\}}(u = a \wedge v = b) = D'_{S'_\ell}(u = a \wedge v = b) = D'_{S'_u}(u = a) \cdot D'_{S'_v}(v = b).$$

The result then follows. ◀

6 All connected components of the correlation graph are small

In this section, we show that all connected components in the correlation graph associated with $\{D_S\}$ are small, which concludes the proof of Theorem 3.

▶ **Theorem 26.** *Assume that the family of distributions $\{D_S\}$ is r-locally consistent and that the hypergraph $H_\mathcal{I}$ is $(\omega(1), k - t/2 + \delta/2)$-expander. Then all connected components in the correlation graph associated with $\{D_S\}$ have size at most $\frac{2k}{\delta}$.*

We will actually prove a slightly more general theorem that we will use to prove LS$_+$ lower bounds in Section 7. Given a hypergraph H, let $G_{\text{bad}}(H)$ be the graph on $[n]$ such that there is an edge between i and j if and only if there exists a bad structure for i and j in H.

▶ **Theorem 27.** *If the hypergraph H is $(\omega(1), k - t/2 + \delta/2)$-expander, then all connected components in $G_{\text{bad}}(H)$ have size at most $\frac{2k}{\delta}$.*

Lemma 24 implies that $G_{\text{bad}}(H_\mathcal{I})$ contains the correlation graph associated with $\{D_S\}$ as a subgraph, so Theorem 26 immediately implies Theorem 27.

Proof of Theorem 27. For any edge e of $G_{\text{bad}}(H)$, we can find a corresponding bad structure W_e. We will say that W_e induces e. Any such bad structure W_e in a $(\omega(1), k-t/2+\delta/2)$-expanding hypergraph satisfies

$$\Gamma(W_e) \leq (k-t)|W_e| + 2 + \frac{k|W_e| - ((k-t)|W_e|+2)}{2} = \left(k - \frac{t}{2}\right)|W_e| + 1. \tag{6.8}$$

The first term upper bounds the number of boundary vertices, the second term counts the endpoints of e, and the last term upper bounds the number of non-boundary vertices. For a connected component in the correlation graph associated with $\{D_S\}$, let $e_1, e_2, \ldots, e_\ell$ be an ordering of edges in the connected component such that $(\bigcup_{j=1}^{i} e_j) \cap e_{i+1}$ is not empty for $i = 1, \ldots, \ell$. That is, $e_1, e_2, \ldots, e_\ell$ is an ordering of the edges in the connected component such that every edge except for the first one is adjacent to some edge preceding it. Let $W_{e_1}, \ldots, W_{e_\ell}$ be corresponding bad structures inducing these edges. Let $T_i = \bigcup_{j=1}^{i} W_{e_j}$ for $i = 1, \ldots, \ell$. While T_i itself is not necessarily a bad structure, we will show that the inequality (6.8) still holds for T_i, i.e.,

$$\Gamma(T_i) \leq \left(k - \frac{t}{2}\right)|T_i| + 1 \tag{6.9}$$

for any $i = 1, \ldots, \ell$. If (6.9) holds, the number of constraints in T_ℓ is at most $\frac{2}{\delta}$; otherwise, expansion is violated. Hence, at most $\frac{2k}{\delta}$ vertices are included in the connected component of the correlation graph associated with $\{D_S\}$.

In the following, we prove (6.9). First, note that $|\Gamma(T_1)| \leq \left(k - \frac{t}{2}\right)|T_1| + 1$ by (6.8). Let $W'_i = W_{e_i} \setminus T_{i-1}$ be the new constraints added at step i. Call any vertex in $\Gamma(T_i) \setminus \Gamma(T_{i-1})$ a new vertex. We will prove that at most $\left(k - \frac{t}{2}\right)|W'_i|$ new vertices are added and this will imply (6.9).

Let n_i be the number of new W'_i-boundary vertices. Then the total number of new vertices is at most

$$n_i + (k|W'_i| - 1 - n_i)/2.$$

The second term upper bounds the number of non-boundary vertices. The -1 comes from the fact that $\Gamma(W'_i)$ must intersect $\Gamma(T_{i-1})$ since e_i must be adjacent to some preceding edge. If $\Gamma(W'_i)$ and $\Gamma(T_{i-1})$ intersect in a boundary vertex, the resulting bound is stronger.

Hence, we would like to upper bound the number n_i of new W'_i-boundary vertices. The number n_i of new W'_i-boundary vertices is at most $(k-t)|W'_i| + 1$ since any new W'_i-boundary vertex must be a new W_{e_i}-boundary vertex and since all but one constraint in W'_i have at most $k-t$ new W_{e_i}-boundary vertices and one constraint in W'_i has at most $k-t+1$ new W_{e_i}-boundary vertices. Hence, the number of new vertices is at most $(k-t)|W'_i| + 1 + (k|W'_i| - 1 - ((k-t)|W'_i|+1))/2 = (k-t/2)|W'_i|$. ◀

From Lemmas 16 and 22 and Theorems 18 and 26, we obtain Theorem 3.

7 LS$_+$ rank lower bounds

In this section, we use the techniques of the previous section to prove positive semidefiniteness of the degree-2 moment matrix of the conditional local distributions $\{D_{S|X=\alpha}\}$. From here, degree lower bounds for the static LS$_+$ proof system andrank lower bounds for LS$_+$ follow easily.

7.1 Positive semidefiniteness of conditional covariance matrices

We define $Y_{X,\alpha}$ to be the conditional covariance matrix of the $\{D_{\{i\}|X=\alpha}\}$ distributions. Formally, define $y_{(X,\alpha)} \in \mathbb{R}^{n+1}$ so that $(y_{X,\alpha})_0 = 1$ and $(y_{X,\alpha})_i = D_{\{i\}|X=\alpha}(i=1)$ for $i \in [n]$. Let $B_{X,\alpha} \in \mathbb{R}^{n \times n}$ have entries $B_{X,\alpha}(i,j) = D_{\{i,j\}|X=\alpha}(i = 1 \wedge j = 1)$. Then define $Y_{X,\alpha} \in \mathbb{R}^{(n+1)\times(n+1)}$ to be

$$\begin{pmatrix} 1 & y_{X,\alpha}^\top \\ y_{X,\alpha} & B_{X,\alpha} \end{pmatrix}.$$

To obtain LS_+ rank lower bounds, we need to show that $Y_{X,\alpha}$ is PSD.

▶ **Lemma 28.** *Let $X \subseteq [n]$ such that $|X| \le \Omega(n^{\frac{\delta}{t-2}})$. For any $\alpha \in \{0,1\}^X$ such that $\mu_C(\alpha) > 0$ for all $C \in \mathcal{C}(X)$, $Y_{X,\alpha}$ is positive semidefinite.*

To prove this lemma, we first show that $Y_{X,\alpha}$ is PSD when $H - X$ has high expansion. Then we show that any $Y_{X,\alpha}$ can expressed as a nonnegative combination of $Y_{\mathsf{Cl}(X),\beta}$'s for $\beta \in \{0,1\}^{\mathsf{Cl}(X)}$; the first step implies that each of these terms is PSD.

We start by generalizing Lemma 24 to conditional distributions. Let $\mathsf{Cl}_X(S)$ be $\mathsf{Cl}(S)$ in the hypergraph $H - X$.

▶ **Lemma 29.** *Let $X \subseteq [n]$ and $\alpha \in \{0,1\}^X$ such that $\mu_C(\alpha) > 0$ for all $C \in \mathcal{C}(X)$. If there is no bad structure for u and v in $H - X$ of size at most $|\mathcal{C}(\mathsf{Cl}_X(\{u,v\}))|$, then u and v are not correlated with respect to $D_{\{u,v\}|X=\alpha}$.*

Proof. First, recall that

$$D_{\{u,v\}|X=\alpha}(u = a \wedge v = b) = \frac{D_{\{u,v\}\cup X}(u = a \wedge v = b \wedge X = \alpha)}{D_X(X = \alpha)}.$$

We will show that $D_{\{u,v\}\cup X}(u = a \wedge v = b \wedge X = \alpha)$ is equal to the product of a term depending on u and a but not v and b and a term depending on v and b but not u and a. From there, the lemma immediately follows.

The proof is essentially the same as that of Lemma 24 above. Starting with $S_0 = \mathcal{C}(\mathsf{Cl}(\{u,v\}))$, we apply the same process except we require that each constraint C_i that we remove satisfies $|(\partial\mathcal{C}(S_{i-1}) \cap C_i) \setminus (\{u,v\} \cup X)| \ge k - t + 1$. At the end of this process, we are left with a set S_ℓ such that $|(\partial\mathcal{C}(S_\ell) \cap C) \setminus (\{u,v\} \cup X)| \le k - t$ for every constraint $C \in \mathcal{C}(S_\ell)$ ($\mathcal{C}(S_\ell)$ could be empty). Let $X_\ell := X \cap S_\ell$ and let $\alpha_\ell = \alpha_{X \cap S_\ell}$. By applying Lemma 25 repeatedly, we see that

$$D_{\{u,v\}\cup X}(u = a \wedge v = b \wedge X = \alpha) = \begin{cases} \frac{1}{q^{|X \setminus S_\ell|+1}} D'_{S_\ell}(u = a \wedge X_\ell = \alpha_\ell) \\ \qquad \text{if } u \in S_\ell, v \notin S_\ell \\ \frac{1}{q^{|X \setminus S_\ell|+1}} D'_{S_\ell}(v = b \wedge X_\ell = \alpha_\ell) \\ \qquad \text{if } v \in S_\ell, u \notin S_\ell \\ \frac{1}{q^{|X \setminus S_\ell|+2}} D'_{S_\ell}(X_\ell = \alpha_\ell) \\ \qquad \text{if } u, v \notin S_\ell \\ \frac{1}{q^{|X \setminus S_\ell|}} D'_{S_\ell}(u = a \wedge v = b \wedge X_\ell = \alpha_\ell) \\ \qquad \text{if } u, v \in S_\ell. \end{cases}$$

In all cases except for the last one, the result follows. In the last case, the assumption that there is no bad structure in $H - X$ implies that the hypergraph induced by $S_\ell - X$ in $H - X$

must be disconnected with u and v in separate connected components with hyperedge sets E_u and E_v just as in the proof of Lemma 24. Each connected component in $H - X$ has a corresponding connected component in H. Let E'_u and E'_v be the sets of hyperedges of the connected components in G corresponding to E_u and E_v in $H - X$. Also, let $S_u = \Gamma(E'_u)$ and $S_v = \Gamma(E'_v)$; note that S_u and S_v might intersect. Let $E_{\text{rest}} = \mathcal{C}(S_\ell) \setminus (E'_u \cup E'_v)$.

The subhypergraph induced by S_ℓ then consists of the hyperedges in E'_u, E'_v, and E_{rest} together with the isolated vertices in $S_{\text{iso}} := \mathcal{S}_\ell \setminus \Gamma(\mathcal{C}(S_\ell))$ not contained in any hyperedge covered by S_ℓ. Define $S_{\text{rest}} := \Gamma(E_{\text{rest}}) \cup S_{\text{iso}}$ to be the vertices in S_ℓ that are either contained in some hyperedge of E_{rest} or are isolated. Let $X_u = X \cap S_u$ and $\alpha_u = \alpha_{X \cap S_u}$. Define X_v, X_{rest}, α_v, and α_{rest} in the same way. We can then write $D'_{S_\ell}(u = a \wedge v = b \wedge X_\ell = \alpha_\ell)$ as

$$D'_{S_u}(u = a \wedge X_u = \alpha_u) \cdot D'_{S_v}(v = b \wedge X_v = \alpha_v) \cdot D'_{S_{\text{rest}}}(X_{\text{rest}} = \alpha_{\text{rest}}).$$

Since $D'_{S_{\text{rest}}}(X_{\text{rest}} = \alpha_{\text{rest}})$ depends only on α, the lemma follows. ◀

▶ Remark. When H does not have high expansion, $\mathsf{Cl}(S)$ is still defined for $S \subseteq [n]$ and Lemma 29 still holds. In this case, it is possible that $|\mathsf{Cl}(S)|$ can no longer be bounded in terms of $|X|$.

Using this lemma, we can prove that $Y_{X,\alpha}$ is PSD when $H - X$ has high enough expansion.

▶ **Lemma 30.** *Let $X \subseteq [n]$ such that $H - X$ is $(\omega(1), k - t/2 + \varepsilon)$-expanding for some constant $\varepsilon > 0$. Then for any $\alpha \in \{0,1\}^X$ with $\mu_X(\alpha) > 0$, $Y_{X,\alpha}$ is positive semidefinite.*

Proof. By Lemma 33, $Y_{X,\alpha}$ is PSD if and only if $Q_{X,\alpha} = B_{X,\alpha} - y_{X,\alpha} y_{X,\alpha}^\top$ is. Note that $Q_{X,\alpha}$ is a principle submatrix of the covariance matrix $\Sigma_{X,\alpha}$ of the $\{D_{S|X=\alpha}\}$ distributions, so it suffices to show that $\Sigma_{X,\alpha}$ is PSD. The conditional distributions $\{D_{S|X=\alpha}\}$ are r-locally consistent for $r = \Omega(n^{\frac{\delta}{t-2}})$ by Corollary 20. Then Lemma 22 implies that $\Sigma_{X,\alpha}$ is PSD if the correlation graph of the $\{D_{S|X=\alpha}\}$ distributions has connected components of size at most r. Lemma 29 implies that correlations under $\{D_{S|X=\alpha}\}$ induce bad structures in $H - X$, and we can apply Theorem 27 to $G_{\text{bad}}(H - X)$ to complete the proof. ◀

Finally, we show that any $Y_{X,\alpha}$ can be expressed as a nonnegative combination of $Y_{\mathsf{Cl}(X),\beta}$'s for $\beta \in \{0,1\}^{\mathsf{Cl}(X)}$.

▶ **Claim 31.**

$$Y_{X,\alpha} = \sum_{\substack{\beta \in \{0,1\}^{\mathsf{Cl}(X)} \\ \beta_X = \alpha}} \frac{D_{\mathsf{Cl}(X)}(\mathsf{Cl}(X) = \beta)}{D_X(X = \alpha)} \cdot Y_{\mathsf{Cl}(X),\beta}.$$

Proof. We will prove that

$$Y_{X,\alpha}(i,j) = \sum_{\substack{\beta \in \{0,1\}^{\mathsf{Cl}(X)} \\ \beta_X = \alpha}} \frac{D_{\mathsf{Cl}(X)}(\mathsf{Cl}(X) = \beta)}{D_X(X = \alpha)} \cdot Y_{\mathsf{Cl}(X),\beta}(i,j).$$

for all $0 \le i, j, \le n$.

Let $i, j \ge 1$. By applying definitions, we see that

$$\begin{aligned} Y_{X,\alpha}(i,j) &= D_{\{i,j\}|X=\alpha}(i = 1 \wedge j = 1) \\ &= \frac{D_{X \cup \{i,j\}}(i = 1 \wedge j = 1 \wedge X = \alpha)}{D_X(X = \alpha)}. \end{aligned}$$

Then, by local consistency, this expression is equal to

$$\frac{1}{D_X(X=\alpha)} \sum_{\substack{\beta \in \{0,1\}^{\mathsf{Cl}(X)} \\ \beta_X = \alpha}} D_{\mathsf{Cl}(X)\cup\{i,j\}}(i = 1 \wedge j = 1 \wedge \mathsf{Cl}(X) = \beta).$$

We can rewrite this as

$$\sum_{\substack{\beta \in \{0,1\}^{\mathsf{Cl}(X)} \\ \beta_X = \alpha}} \frac{D_{\mathsf{Cl}(X)}(\mathsf{Cl}(X) = \beta)}{D_X(X=\alpha)} \cdot \frac{D_{\mathsf{Cl}(X)\cup\{i,j\}}(i = 1 \wedge j = 1 \wedge \mathsf{Cl}(X) = \beta)}{D_{\mathsf{Cl}(X)}(\mathsf{Cl}(X) = \beta)}.$$

Using the definitions of conditional local distributions and $Y_{X,\alpha}(i,j)$ completes the proof for $i, j \geq 1$: the above expression is equal to

$$\sum_{\substack{\beta \in \{0,1\}^{\mathsf{Cl}(X)} \\ \beta_X = \alpha}} \frac{D_{\mathsf{Cl}(X)}(\mathsf{Cl}(X) = \beta)}{D_X(X=\alpha)} \cdot Y_{\mathsf{Cl}(X),\beta}(i,j).$$

When i or j is equal to 0, an essentially identical argument can be used. ◀

Lemma 28 follows immediately from Lemma 30 and Claim 31.

7.2 Rank lower bounds for static LS_+ and LS_+

We now use the results of the previous section to prove Theorem 5, a lower bound on the degree of any static LS_+ refutation. The proof is essentially the same as that of [30, Theorem 3.27], which is the special case of P being pairwise uniform. This will immediately imply Theorem 4, the rank lower bound for LS_+.

Proof of Theorem 5. Assume that the rank of any standard, non-static LS_+ proof is at least r; by Theorem 4, $r = \Omega(n^{\frac{\varepsilon}{t-2}})$. Assume for a contradiction that there exists a static LS_+ refutation of degree $r - k$. Recall that a static LS_+ refutation has the form

$$\sum_{\ell=1}^{r} w_\ell q_\ell(x) \phi_{I_\ell, J_\ell}(x) = -1, \tag{7.10}$$

where $w_\ell \geq 0$, each $q_\ell(x)$ is either an axiom or the square of a linear form, and

$$\phi_{I,J}(x) = \prod_{i \in I} x_i \prod_{j \in J} (1 - x_j).$$

By Theorem 18, we know that there exist r-consistent local distributions $\{D_S\}$; let $\widetilde{\mathbb{E}}[\cdot]$ be the corresponding rank-r SA pseudoexpectation. In addition, Corollary 20 states that there also exist conditional $(r - |X|)$-consistent local distributions $\{D_{S|X=\alpha}\}$ for any α such that $\mu(\alpha_C) > 0$ for all $C \in \mathcal{C}(X)$,

We will derive a contradiction by applying the operator $\widetilde{\mathbb{E}}[\cdot]$ to both sides of (7.10). Specifically, we will show that if the degree of each term is at most $r - k$, then $\widetilde{\mathbb{E}}[q_\ell(x)\phi_{I_\ell,J_\ell}(x)] \geq 0$. Applying $\widetilde{\mathbb{E}}[\cdot]$ to the left hand side of (7.10) gives a value at least 0, while the right hand side is -1. To show that $\widetilde{\mathbb{E}}[q_\ell(x)\phi_{I_\ell,J_\ell}(x)] \geq 0$, we consider two cases.

Case 1. q_ℓ is an axiom. If q_ℓ is an axiom, then the number of variables in the expression $q_\ell(x)\phi_{I_\ell,J_\ell}(x)$ is $|I_\ell \cup J_\ell \cup \mathrm{supp}(q_\ell)| \leq r - k + k = r$. We also know that $q_\ell(x)\phi_{I_\ell,J_\ell}(x) \geq 0$ for all $x \in \{0,1\}^n$. The definition of rank-r SA pseudodistributions then implies that $\widetilde{\mathbb{E}}[q_\ell(x)\phi_{I_\ell,J_\ell}(x)] \geq 0$.

Case 2. q_ℓ is the square of a linear form. In this case, the result follows almost immediately from Lemma 28. Write $q_\ell(x)$ as follows:

$$q_\ell(x) = \left(a_0 + \sum_{i\in[n]} a_i x_i\right)^2 = \sum_{i,j\in[n]} a_i a_j x_i x_j + 2a_0 \sum_{i\in[n]} a_i x_i + a_0^2.$$

Also, let $A_\ell = I_\ell \cup J_\ell$ and define $\beta \in \{0,1\}^{|A_\ell|}$ so that $\phi_{I_\ell,J_\ell}(x) = 1$ if and only if $x_{A_\ell} = \beta$. Then we have the following calculation:

$$\begin{aligned}
\widetilde{\mathbb{E}}[q_\ell(x)\phi_{I_\ell,J_\ell}(x)] &= \widetilde{\mathbb{E}}[q_\ell(x)\cdot \mathbf{1}(A_\ell = \beta)] \\
&= \sum_{i,j\in[n]} a_i a_j \widetilde{\mathbb{E}}[x_i x_j \cdot \mathbf{1}(A_\ell = \beta)] + 2a_0 \sum_{i\in[n]} a_i \widetilde{\mathbb{E}}[x_i \cdot \mathbf{1}(A_\ell = \beta)] \\
&\qquad + a_0^2 \widetilde{\mathbb{E}}[\mathbf{1}(A_\ell = \beta)] \\
&= \sum_{i,j\in[n]} a_i a_j D_{A_\ell\cup\{i,j\}}(A_\ell = \beta) Y_{A_\ell,\beta}(i,j) \\
&\qquad + 2a_0 \sum_{i\in[n]} a_i D_{A_\ell\cup\{i,j\}}(A_\ell = \beta) Y_{A_\ell,\beta}(i,0) + a_0^2 D_{A_\ell}(A_\ell = \beta) \\
&= D_{A_\ell}(A_\ell = \beta) a^\top (Y_{A_\ell,\beta}) a \quad \text{by } r\text{-local consistency of } \{D_S\} \\
&\geq 0 \qquad \text{by Lemma 28.}
\end{aligned}$$
◀

Theorem 4, our rank lower bound for LS$_+$ refutations, follows immediately from Theorem 5 and the following fact.

▶ **Fact 32.** *If there exists a rank-r LS$_+$ refutation of a set of axioms A, then there exists a static LS$_+$ refutation of A with rank at most r.*

Proof. Let R be a rank-r LS$_+$ refutation. We look at R as a DAG in which each node is the application of some inference rule, the root is $-1 \geq 0$, and the leaves are axioms or applications of the rule $P(x)^2 \geq 0$ for P with degree at most 1. Starting from the root $-1 \geq 0$ and working back to the axioms, we can substitute in the premises of each inference to get an expression $Q(x) = -1$. Since R has rank r, each path in r has at most r multiplications by a term of the form x_i or $(1 - x_i)$ and $Q(x) = -1$ must be a valid static LS$_+$ refutation of rank at most r. ◀

Acknowledgments. The first-named author would like to thank Osamu Watanabe for his encouragement. The second-named author would like to thank Anupam Gupta and Ryan O'Donnell for several helpful discussions.

References

1 Sarah R. Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 689–708, 2015.

2 Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, pages 171–180, 2010.

3 Boaz Barak. Lecture 1 – Introduction. Notes from course "Sum of Squares upper bounds, lower bounds, and open questions".

4 Boaz Barak, Fernando G. S. L. Brandão, Aram W. Harrow, Jonathan Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, Sum-of-Squares Proofs, and their Applications. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 307–326, 2012.

5 Boaz Barak, Siu On Chan, and Pravesh Kothari. Sum of squares lower bounds from pairwise independence. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pages 97–106, 2015.

6 Boaz Barak, Guy Kindler, and David Steurer. On the Optimality of Semidefinite Relaxations for Average-Case and Generalized Constraint Satisfaction. In *Proceedings of the 4th Innovations in Theoretical Computer Science conference*, 2013.

7 Boaz Barak and Ankur Moitra. Tensor Prediction, Rademacher Complexity and Random 3-XOR. *CoRR*, abs/1501.06521, 2015. URL: `http://arxiv.org/abs/1501.06521`.

8 Eli Ben-Sasson and Yonatan Bilu. A gap in average proof complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 9(3), 2002.

9 Siavosh Benabbas, Konstantinos Georgiou, Avner Magen, and Madhur Tulsiani. SDP gaps from pairwise independence. *Theory of Computing*, 8:269–289, 2012.

10 Joshua Buresh-Oppenheim, Nicola Galesi, Shlomo Hoory, Avner Magen, and Toniann Pitassi. Rank bounds and integrality gaps for cutting planes procedures. *Theory of Computing*, 2:65–90, 2006. `doi:10.4086/toc.2006.v002a004`.

11 Siu On Chan, James R. Lee, Prasad Raghavendra, and David Steurer. Approximate constraint satisfaction requires large LP relaxations. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, pages 350–359, 2013.

12 Amin Coja-Oghlan, Andreas Goerdt, and André Lanka. Strong Refutation Heuristics for Random k-SAT. In Klaus Jansen, Sanjeev Khanna, José D.P. Rolim, and Dana Ron, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 3122 of *Lecture Notes in Computer Science*, pages 310–321. Springer Berlin Heidelberg, 2004. `doi:10.1007/978-3-540-27821-4_28`.

13 A Crisanti, L Leuzzi, and G Parisi. The 3-SAT problem with large number of clauses in the ∞-replica symmetry breaking scheme. *Journal of Physics A: Mathematical and General*, 35(3):481, 2002.

14 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 441–448, 2014.

15 Sanjeeb Dash. *On the Matrix Cuts of Lovász and Schrijver and their use in Integer Programming*. PhD thesis, Rice University, 2001.

16 Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large k. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pages 59–68, 2015.

17 Uriel Feige. Relations Between Average Case Complexity and Approximation Complexity. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 534–543, 2002.

18 Uriel Feige and Eran Ofek. Easily refutable subformulas of large random 3CNF formulas. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming*, volume 3142 of *Lecture Notes in Comput. Sci.*, pages 519–530. Springer, Berlin, 2004.

19 Vitaly Feldman, Will Perkins, and Santosh Vempala. On the Complexity of Random Satisfiability Problems with Planted Solutions. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pages 77–86, 2015.

20 Joel Friedman, Andreas Goerdt, and Michael Krivelevich. Recognizing more unsatisfiable random k-SAT instances efficiently. *SIAM J. Comput.*, 35(2):408–430, 2005. `doi:10.1137/S009753970444096X`.

21 Dima Grigoriev. Linear lower bound on degrees of Positivstellensatz calculus proofs for the parity. *Theoretical Computer Science*, 259(1-2):613–622, 2001.

22 Dima Grigoriev, Edward A. Hirsch, and Dmitrii V. Pasechnik. Complexity of semi-algebraic proofs. In *Proceedings of the 19th International Symposium on Theoretical Aspects of Computer Science*, pages 419–430, 2002.

23 Arist Kojevnikov and Dmitry Itsykson. Lower Bounds of Static Lovász-Schrijver Calculus Proofs for Tseitin Tautologies. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, 2006.

24 James R. Lee, Prasad Raghavendra, and David Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pages 567–576, 2015.

25 László Lovász and Alexander Schrijver. Cones of Matrices and Set-Functions and 0-1 Optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

26 Ryan O'Donnell and David Witmer. Goldreich's PRG: Evidence for near-optimal polynomial stretch. In *Proceedings of the 29th Annual Conference on Computational Complexity*, pages 1–12, 2014.

27 Prasad Raghavendra. Optimal Algorithms and Inapproximability Results for Every CSP? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 245–254, 2008.

28 Grant Schoenebeck. Linear Level Lasserre Lower Bounds for Certain k-CSPs. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 593–602, 2008.

29 Hanif Sherali and Warren Adams. A Hierarchy of Relaxations between the Continuous and Convex Hull Representations for Zero-One Programming Problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

30 Madhur Tulsiani and Pratik Worah. LS_+ lower bounds from pairwise independence. In *Proceedings of the 28th Annual Conference on Computational Complexity*, pages 121–132, 2013.

31 Martin J. Wainwright and Michael I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*, volume 1. Now Publishers Inc., Hanover, MA, USA, January 2008. `doi:10.1561/2200000001`.

A Proofs from Section 2.4

▶ **Fact 15.** *$(s, k-d)$-expansion implies $(s, k-2d)$-boundary expansion.*

Proof. Let S be a set of at most s hyperedges. Each of the vertices in $\Gamma(S)$ is either a boundary vertex that appears in exactly one hyperedge or it appears in two or more hyperedges, so $|\Gamma(S)| \leq |\partial S| + \frac{1}{2}(|k|S| - |\partial S|)$. Therefore, we can write

$$|\partial S| \geq 2|\Gamma(S)| - k|S| \geq (k-2d)|S|,$$

where the second inequality follows the expansion assumption. ◀

▶ **Lemma 16.** *Fix $\delta > 0$. With high probability, a set of $m \leq \Omega(n^{t/2-\delta})$ constraints chosen uniformly at random is both $\left(n^{\frac{\delta}{t-2}}, k - \frac{t}{2} + \frac{\delta}{2}\right)$-expanding and $\left(n^{\frac{\delta}{t-2}}, k - t + \delta\right)$-boundary expanding.*

Proof. By Fact 15, it suffices to show that a random instance is $\left(n^{\frac{\delta}{t-2}}, k - \frac{t}{2} + \frac{\delta}{2}\right)$-expanding. We give the proof of [26], which is essentially the same as that of [9].

We want to upper bound the probability that any set of r hyperdges with $r \leq n^{\frac{\delta}{t-2}}$ contains less than $r(k - \frac{t}{2} + \frac{\delta}{2})$ vertices. Fix an r-tuple of edges T; this is a tuple of indices in $[m]$ representing the indices of the hyperedges in T. We wish to upper bound $\Pr[|\Gamma(T)| \leq v]$; we can do this with the quantity

$$\frac{(\#\text{ sets } S \text{ of } v \text{ vertices}) \cdot (\#\text{ sets of } r \text{ edges contained in } S)}{(\#\text{ of ways of choosing } r \text{ edges})}.$$

Taking a union bound over all tuples of size r, we see that

$$\Pr[|\Gamma(S)| \leq v\ \forall S \text{ s.t. } |S| = r] \leq r!\binom{m}{r} \cdot \frac{\binom{n}{v}\binom{k!\binom{v}{k}}{r}}{(k!\binom{n}{k})^r}.$$

Simplifying and applying standard approximations, we get that

$$\Pr[|\Gamma(S)| \leq v\ \forall S \text{ s.t. } |S| = r] \leq e^{(2+k)r+v} v^{kr-v} r^{-r} n^{v-kr} m^r.$$

Set $v = \lfloor r(k - \frac{t}{2} + \frac{\delta}{2}) \rfloor$ and simplify to get

$$\Pr[|\Gamma(S)| < r(k - \frac{t}{2} + \frac{\delta}{2})\ \forall S \text{ s.t. } |S| = r] \leq (C(k,t) m n^{-(t/2-\delta/2)} r^{t/2-1-\delta/2})^r$$

Then set $m = n^{t/2-\delta}$ and take a union bound over all choices of r to get that

$$\begin{aligned}
&\Pr[H_{\mathcal{I}} \text{ not } (n^{\frac{\delta}{t-2}}, k - \frac{t}{2} + \frac{\delta}{2})\text{-expanding}] \leq \sum_{r=1}^{\lfloor n^{\delta/(t-2)} \rfloor} (C(k,t) n^{-\delta/2} r^{t/2-1-\delta/2})^r \\
&= \sum_{r=1}^{\lceil \log n \rceil} (C(k,t) n^{-\delta/2} r^{t/2-1-\delta/2})^r + \sum_{r=\lceil \log n \rceil+1}^{\lfloor n^{\delta/(t-2)} \rfloor} (C(k,t) n^{-\delta/2} r^{t/2-1-\delta/2})^r \\
&\leq 2C(k,t) n^{-\delta/2} (\log n)^{t/2-\delta/2} + n^{\frac{\delta}{t-2}} (C(k,t) n^{-\delta/2} (n^{\frac{\delta}{t-2}})^{t/2-1-\delta/2})^{\log n} \\
&= O(n^{-\delta/3}).
\end{aligned}$$

◀

B Equivalence between PSDness of the degree-2 moment matrix and the covariance matrix

▶ **Lemma 33.**

$$\begin{pmatrix} 1 & w^T \\ w & B \end{pmatrix} \text{ is PSD} \iff B - ww^T \text{ is PSD.}$$

Proof.

$$\begin{aligned}
\begin{pmatrix} 1 & w^T \\ w & B \end{pmatrix} \text{ is PSD} &\iff \left((v_0\, v) \begin{pmatrix} 1 & w^T \\ w & B \end{pmatrix} (v_0\, v)^T \geq 0\ \forall v_0 \in \mathbb{R},\ v \in \mathbb{R}^{nq} \right) \\
&\iff \left(v_0^2 + 2\langle w, v \rangle v_0 + \langle Bv, v \rangle \geq 0\ \forall v_0 \in \mathbb{R},\ v \in \mathbb{R}^{nq} \right) \\
&\iff \left((v_0 + \langle w, v \rangle)^2 - \langle w, v \rangle^2 + \langle Bv, v \rangle \geq 0\ \forall v_0 \in \mathbb{R},\ v \in \mathbb{R}^{nq} \right) \\
&\iff \left(-\langle w, v \rangle^2 + \langle Bv, v \rangle \geq 0\ \forall v \in \mathbb{R}^{nq} \right) \\
&\iff \left(v(B - ww^T)v^T \geq 0\ \forall v \in \mathbb{R}^{nq} \right) \\
&\iff B - ww^T \text{ is PSD.}
\end{aligned}$$

◀

▶ **Lemma 10.** *M is PSD if and only if Σ is PSD.*

Proof. We rewrite M as

$$\begin{pmatrix} 1 & w^T \\ w & B \end{pmatrix},$$

where w is a vector whose (i,a)-element is $p_{\{i\}}(x_i = a)$ for $i \in [n]$ and $a \in [q]$, where B is a matrix whose $((i,a),(j,b))$-element is $p_{\{i,j\}}(x_i = a \wedge x_j = b)$ for $i,j \in [n]$ and $a,b \in [q]$. From Lemma 33, we know that

$$\begin{pmatrix} 1 & w^T \\ w & B \end{pmatrix} \text{ is PSD if and only if } B - ww^T \text{ is PSD.}$$

Observe that $B - ww^T$ is equal to the covariance matrix Σ. ◀

C Proofs from Section 2.5

▶ **Lemma 17.** *If $H_{\mathcal{I}}$ is (s_1,e_1)-expanding and S is a set of variables such that $|S| < (e_1-e_2)s_1$ for some $e_2 \in (0,e_1)$, then there exists a set $\mathsf{Cl}(S) \subseteq [n]$ such that $S \subseteq \mathsf{Cl}(S)$ and $H_{\mathcal{I}} - \mathsf{Cl}(S)$ is (s_2,e_2)-expanding with $s_2 \geq s_1 - \frac{|S|}{e_1-e_2}$ and $\mathsf{Cl}(S) \leq \frac{e_1}{e_1-e_2}|S|$.*

Proof. We calculate $\mathsf{Cl}(S)$ using the closure algorithm of [9, 30]:

Input: An (s_1,e_1)-expanding instance $\mathcal{I}$, $e_2 \in (0,e_1)$, a tuple $S = (x_1,\dots,x_u) \in [n]^u$ such that $u < (e_1-e_2)s_2$.
Output: The closure $\mathsf{Cl}(S)$.

```
Set $\mathsf{Cl}(S) \leftarrow \emptyset$ and $s_2 \leftarrow s_1$.
for $i = 1,\dots,u$
    $\mathsf{Cl}(S) \leftarrow \mathsf{Cl}(S) \cup \{x_i\}$
    if $H_{\mathcal{I}} - \mathsf{Cl}(S)$ is not $(s_2,e_2)$-expanding, then
        Find largest set of constraints $M_i$ in $H_{\mathcal{I}} - \mathsf{Cl}(S)$ such that
        $|M_i| \leq s_2$ and $|\Gamma(M_i)| \leq e_2|M_i|$. Break ties by lexicographic order.
        $\mathsf{Cl}(S) \leftarrow \mathsf{Cl}(S) \cup \Gamma(M_i)$
        $s_2 \leftarrow s_2 - |M_i|$
return $\mathsf{Cl}(S)$
```

It is clear from the statement of the algorithm that $S \subseteq \mathsf{Cl}(S)$. We need to show that $H_{\mathcal{I}} - \mathsf{Cl}(S)$ is (s_2,e_2)-expanding, that $s_2 \geq s_1 - \frac{|S|}{e_1-e_2}$, and that $\mathsf{Cl}(S) \leq \frac{e_1}{e_1-e_2}|S|$. We give the proof of [9].

1. $H_{\mathcal{I}} - \mathsf{Cl}(S)$ is (s_2,e_2)-expanding
   We will show that at every step of the algorithm $H_{\mathcal{I}} - \mathsf{Cl}(S)$ is (s_2,e_2)-expanding. Say we are in step i and that $H_{\mathcal{I}} - (\mathsf{Cl}(S) \cup \{x_i\})$ is not (s_2,e_2)-expanding; if it were (s_2,e_2)-expanding, we would be done. Let M_i be the largest set of hyperedges in $H_{\mathcal{I}} - \mathsf{Cl}(S)$ such that $|M_i| \leq s_2$ and $|\Gamma(M_i)| \leq e_2|M_i|$. We need to show that $H_{\mathcal{I}} - (\mathsf{Cl}(S) \cup \{x_i\} \cup \Gamma(M_i))$ is $(\zeta - |M_i|, e_2)$-expanding.
   To see this, assume for a contradiction that there exists a set of hyperedges M' in $H_{\mathcal{I}} - (\mathsf{Cl}(S) \cup \{x_i\} \cup \Gamma(M_i))$ such that $M' \leq s_2 - |M_i|$ and $|\Gamma(M')| < e_2|M'|$. Consider $M_i \cup M'$. Note that M_i and M' are disjoint, so $|M_i \cup M'| \leq s_2$. Also, $|\Gamma(M_i \cup M')| \leq e_2|M_i| + e_2|M'| = e_2|M_i \cup M'|$, contradicting the maximality of M_i.
2. $s_2 \geq s_1 - \frac{|S|}{e_1-e_2}$
   Consider the set $M = \bigcup_{i=1}^{u} M_u$. First, note that $|M| = s_1 - s_2$, so $|\Gamma(M)| \geq e_1(s_1 - s_2)$

by expansion of $H_{\mathcal{I}}$. Second, each element of $\Gamma(M) - S$ occurs in exactly one of the M_i's and each M_i has expansion at most e_2. Using these two observations, we see that

$$e_1(s_1 - s_2) \leq |\Gamma(M)| \leq |S| + \sum_{i=1}^{u} e_2|M_i| = |S| + e_2(s_1 - s_2).$$

This implies the claim.

3. $\mathsf{Cl}(S) \leq \frac{e_1}{e_1-e_2}|S|$
Observe that $\mathsf{Cl}(S) = S \cup \bigcup_{i=1}^{u} \Gamma(M_i)$. Also, every M_i has expansion at most e_2. Therefore, we have that

$$\begin{aligned} |\mathsf{Cl}(S)| &\leq |S| + \sum_{i=1}^{u} |\Gamma(M_i)| \\ &\leq |S| + e_2 \sum_{i=1}^{u} |M_i| \\ &\leq |S| + \frac{e_2|S|}{e_1 - e_2} \\ &= \left(\frac{e_1}{e_1 - e_2}\right)|S|, \end{aligned}$$

where we used that $\sum_{i=1}^{u} |M_i| = s_1 - s_2$ and $s_2 \geq s_1 - \frac{|S|}{e_1-e_2}$. ◀

▶ **Theorem 18.** *For a random instance $\mathcal{I}$ with $m \leq \Omega(n^{t/2-\varepsilon})$, the family of distributions $\{D_S\}_{|S|\leq r}$ is r-locally consistent for $r = n^{\frac{\varepsilon}{t-2}}$ and is supported on satisfying assignments.*

To prove the theorem, we will use the following lemma, which says that the local distributions on D'_S and D'_T with $S \subseteq T$ are consistent if $H_{\mathcal{I}} - T$ has high boundary expansion.

▶ **Lemma 34.** *Let P be a $(t-1)$-wise uniform supporting predicate, let $\mathcal{I}$ be an instance of* CSP(P)*, and let $S \subseteq T$ be sets of variables. If $H_{\mathcal{I}}$ and $H_{\mathcal{I}} - S$ are $(r, k-t+\varepsilon)$-boundary expanding for some $\varepsilon > 0$ and $\mathcal{C}(T) \leq r$, then for any $\alpha \in [q]^S$,*

$$D'_S(\alpha) = \sum_{\substack{\beta\in[q]^T \\ \beta_S=\alpha}} D'_T(\beta).$$

First, we will use this lemma to prove Theorem 18.

Proof of Theorem 18. Let $S \subseteq T$ be sets of variables with $|T| \leq r$. Consider $U = \mathsf{Cl}(S) \cup \mathsf{Cl}(T)$. We will show that both D_S and D_T are consistent with U and therefore must themselves be consistent. Observe that $|\mathsf{Cl}(S)|$ and $|\mathsf{Cl}(T)|$ are at most $\frac{2kr}{\varepsilon}$, so $|U| \leq \frac{4kr}{\varepsilon}$. Towards applying Lemma 34, we will first show that $|\mathcal{C}(U)| \leq \frac{8r}{\varepsilon}$. Assume for a contradiction that C is a subset of $\mathcal{C}(U)$ of size $\frac{8r}{\varepsilon}$. Then

$$\frac{|\Gamma(C)|}{|C|} \leq \frac{|U|}{|C|} = \frac{4kr/\varepsilon}{8r/\varepsilon} = \frac{k}{2} < k - \frac{t}{2} + \delta,$$

which violates expansion.

We know that $H_{\mathcal{I}} - \mathsf{Cl}(T)$ and $H_{\mathcal{I}} - \mathsf{Cl}(S)$ are $(r, k-t+\varepsilon)$-boundary expanding for some $\varepsilon > 0$. We can then apply Lemma 34 twice with sets $\mathsf{Cl}(S) \subseteq U$ and $\mathsf{Cl}(T) \subseteq U$ to see that

$$\begin{aligned} D_S(\alpha) = \sum_{\substack{\beta \in [q]^{\mathsf{Cl}(S)} \\ \beta_S = \alpha}} D'_{\mathsf{Cl}(S)}(\beta) &= \sum_{\substack{\gamma \in [q]^U \\ \gamma_S = \alpha}} D'_U(\gamma) \\ &= \sum_{\substack{\beta' \in [q]^{\mathsf{Cl}(T)} \\ \beta'_S = \alpha}} D'_{\mathsf{Cl}(T)}(\beta') = \sum_{\substack{\alpha' \in [q]^T \\ \alpha'_S = \alpha}} D_T(\alpha'). \end{aligned}$$ ◀

Now we prove Lemma 34.

Proof of Lemma 34. We follow the proof of Benabbas et al. [9]. Let $\mathcal{C}(T) \setminus \mathcal{C}(S) = \{C_1, \ldots, C_u\}$ and, for a constraint C, let $\sigma(C)$ be the variables in the support of C. First, observe that

$$\begin{aligned} Z'_T \sum_{\substack{\beta \in [q]^T \\ \beta_S = \alpha}} D'_T(\beta) &= \sum_{\gamma \in [q]^{T \setminus S}} \prod_{C \in \mathcal{C}(T)} \mu_C((\alpha, \gamma)) \\ &= \left(\prod_{C \in \mathcal{C}(S)} \mu_C(\alpha) \right) \sum_{\gamma \in [q]^{S \setminus T}} \prod_{i=1}^{u} \mu_{C_i}((\alpha, \gamma)) \\ &= (Z'_S D'_S(\alpha)) \sum_{\gamma \in [q]^{S \setminus T}} \prod_{i=1}^{u} \mu_{C_i}((\alpha, \gamma)) \end{aligned}$$

To finish the proof, we will need the following claim.

▶ **Claim 35.** *There exists an ordering $(C_{i_1}, \ldots, C_{i_u})$ of constraints of $\mathcal{C}(T) \setminus \mathcal{C}(S)$ and a partition $V_1, \cdots, V_u, V_{u+1}$ of variables of $T \setminus S$ such that for all $j \le u$ the following hold.*

1. *$V_j \subseteq \sigma(C_{i_j})$.*
2. *$|V_j| \ge k - t + 1$*
3. *V_j does not intersect $\sigma(C_{i_l})$ for any $l > j$. That is, $V_j \cap \bigcup_{l>j} \sigma(C_{i_l}) = \emptyset$.*

Proof of Claim 35. We will find the sets V_j by repeatedly using $(r, k-t+\delta)$-boundary expansion of $H_{\mathcal{I}} - S$. Let $Q_1 = \mathcal{C}(T) \setminus \mathcal{C}(S)$. We know that $|Q_1| \le r$, so boundary expansion of $H_{\mathcal{I}} - S$ implies that $|\partial(Q_1) \setminus S| \ge (k-t+\delta)|Q_1|$. There must exist a constraint $C_j \in Q_1$ with at least $k-t+1$ boundary variables in $H_{\mathcal{I}} - S$; i.e., $|\sigma(C_j) \cap (\partial(Q_1) \setminus S)| \ge k-2$. We then set $V_1 = \sigma(C_j) \cap (\partial(Q_1) \setminus S)$ and $i_1 = j$. Let $Q_2 = Q_1 \setminus C_j$. We apply the same process $u-1$ more times until Q_l is empty and then set $V_{u+1} = (T \setminus S) \setminus (\bigcup_{j=1}^{u} V_j)$. We remove constraint C_{i_l} at every step and $F_l \subseteq \sigma(C_{i_l})$, so it holds that $V_j \cap \bigcup_{l>j} \sigma(C_{i_l}) = \emptyset$. ◀

Using the claim, we can write $\sum_{\gamma \in [q]^{S \setminus T}} \prod_{i=1}^{u} \mu_{C_i}((\alpha, \gamma))$ as

$$\sum_{\gamma_{u+1} \in [q]^{V_{u+1}}} \sum_{\gamma_u \in [q]^{V_u}} \mu_{C_u}(\gamma'_u) \sum_{\gamma_{u-1} \in [q]^{V_{u-1}}} \mu_{C_{u-1}}(\gamma'_{u-1}) \cdots \sum_{\gamma_1 \in [q]^{V_1}} \mu_{C_1}(\gamma'_1),$$

where each γ'_j depends on α and γ_l with $l \ge j$ but does not depend on γ_l with $l < j$. We will evaluate this sum from right to left. We know that each V_j contains at least $k-t+1$ elements, so $(t-1)$-wise uniformity of μ implies that $\sum_{\gamma_j \in [q]^{V_j}} \mu_{C_j}(\gamma'_j) = q^{-(k-|V_j|)}$. Applying this repeatedly, we see that

$$\sum_{\gamma \in [q]^{S \setminus T}} \prod_{i=1}^{u} \mu_{C_i}((\alpha, \gamma)) = q^{-(ku - \sum_{j=1}^{u+1} |V_j|)} = q^{|T \setminus S| - k|\mathcal{C}(T) \setminus \mathcal{C}(S)|}.$$

Plugging this quantity into the above calculation, we obtain

$$Z'_T \sum_{\substack{\beta \in [q]^T \\ \beta|_S = \alpha}} D'_T(\beta) = Z'_S D'_S(\alpha) q^{|T \setminus S| - k|\mathcal{C}(T) \setminus \mathcal{C}(S)|}.$$

Since $H_{\mathcal{I}}$ has $(r, k - t + \delta)$-boundary expansion for some $\delta > 0$, we can set $S = \emptyset$ to get that $Z'_T = q^{|T| - k|\mathcal{C}(T)|}$. Similarly, $Z'_S = q^{|S| - k|\mathcal{C}(S)|}$. Plugging these two quantities in completes the proof. ◀

▶ **Lemma 19.** *Let $X \subseteq [n]$ and let $\{D_S\}$ be a family of r-locally consistent distributions for sets $S \subseteq [n]$ such that $S \cap X = \emptyset$ and $|S \cup X| \leq r$. Then the family of conditional distributions $\{D_S(\cdot | X = \alpha)\}$ is $(r - |X|)$-locally consistent for any $\alpha \in \{0,1\}^X$ such that $\mu(\alpha|_C) > 0$ for all constraints in $\mathcal{C}(X)$.*

Proof. Tulsiani and Worah proved this lemma and we will use their proof [30]. Let $S \subseteq T$ and $|T \cup X| \leq r$. Let β be any assignment to S. Then local consistency of the $\{D_S\}$ measures implies that $D_{S \cup X}(S = \beta \wedge X = \alpha) = D_{T \cup X}(S = \beta \wedge X = \alpha)$ and $D_{S \cup X}(X = \alpha) = D_{T \cup X}(X = \alpha)$. We therefore have that

$$\begin{aligned} D_{S|X=\alpha}(S = \beta) &= \frac{D_{S \cup X}(S = \beta \wedge X = \alpha)}{D_{S \cup X}(X = \alpha)} \\ &= \frac{D_{T \cup X}(S = \beta \wedge X = \alpha)}{D_{T \cup X}(X = \alpha)} = D_{T|X=\alpha}(S = \beta). \end{aligned}$$ ◀

D Equivalence of SA, SA$_+$, and static LS$_+$ tightenings of linear and degree-k relaxations of CSP(P)

▶ **Lemma 12.** *Let $r \geq k$. Then the following statements hold.*

1. $\mathrm{SA}^r(R_{\mathcal{I}}) = \mathrm{SA}^r(L_{\mathcal{I}})$.
2. $\mathrm{SA}^r_+(R_{\mathcal{I}}) = \mathrm{SA}^r_+(L_{\mathcal{I}})$.
3. $\mathrm{StaticLS}^r_+(R_{\mathcal{I}}) = \mathrm{StaticLS}^r_+(L_{\mathcal{I}})$

Proof. The proof is the same for SA, SA$_+$, and static LS$_+$. We use the notation introduced in Section 2.2.1. For $f \in \{0,1\}^k$ and $z \in [0,1]^k$, let $P'_f(z) = \sum_{i=1}^k z^{(f_i)}$. Let $(c, S) \in \mathcal{I}$ be any constraint. Let $\widetilde{\mathbb{E}}[\cdot]$ be any r-round SA pseudoexpectation. We begin by making a couple of observations. Let q be an arbitrary multilinear polynomial satisfying the following conditions.

1. $q(x) \geq 0$ for all $x \in \{0,1\}^n$.
2. $q(x) \cdot (P'(x_S^{(c)}) - 1)$ depends on at most r variables. Equivalently, $q(x) \cdot (P'_f(x_S^{(c)}) - 1)$ depends on at most r variables for all $f \in F$.

First, note that

$$P'(x_S^{(c)}) - 1 = \sum_{f \in F} 1_{\{x_S = f\}}(x) \cdot (P'_f(x_S^{(c)}) - 1). \tag{D.11}$$

This implies that

$$\widetilde{\mathbb{E}}[p(x) \cdot (P'(x_S^{(c)}) - 1)] = \sum_{f \in F} \widetilde{\mathbb{E}}[p(x) \cdot 1_{\{x_S = f\}}(x) \cdot (P'_f(x_S^{(c)}) - 1)]. \tag{D.12}$$

Second, we see that $-q(x)\cdot 1_{\{x_S=f\}}(x)\cdot(P'_f(x_S^{(c)})-1)\geq 0$ for all $x\in\{0,1\}^n$ for all $(c,S)\in\mathcal{I}$, and for all $f\in F$. Since Condition 2 implies that $-q(x)\cdot 1_{\{x_S=f\}}(x)\cdot(P'_f(x_S^{(c)})-1)$ depends on at most r variables and $\widetilde{\mathbb{E}}$ is a degree-r SA pseudexpectation,

$$\widetilde{\mathbb{E}}[q(x)\cdot 1_{\{x_S=f\}}(x)\cdot(P'_f(x_S^{(c)})-1)]\leq 0. \tag{D.13}$$

Now assume that $\widetilde{\mathbb{E}}[\cdot]$ satisfies

$$\widetilde{\mathbb{E}}[p(x)\cdot(P'_f(x_S^{(c)})-1)]\geq 0 \tag{D.14}$$

for all $f\in F$ and for all multilinear polynomials p satisfying conditions 1 and 2. We want to show that $\widetilde{\mathbb{E}}[q(x)\cdot(P'(x_S^{(c)})-1)]=0$ for all multilinear q satisfying conditions 1 and 2. Since $q(x)\cdot 1_{\{x_S=f\}}(x)$ is nonnegative and $q(x)\cdot 1_{\{x_S=f\}}(x)\cdot(P'_f(x_S^{(c)})-1)$ depends on at most r variables, (D.14) implies that $\widetilde{\mathbb{E}}[q(x)\cdot 1_{\{x_S=f\}}(x)\cdot(P'_f(x_S^{(c)})-1)]\geq 0$. Together with (D.13), this implies that $\widetilde{\mathbb{E}}[q(x)\cdot 1_{\{x_S=f\}}(x)\cdot(P'_f(x_S^{(c)})-1)]=0$ and the result follows from (D.12).

For the other direction, assume that $\widetilde{\mathbb{E}}[\cdot]$ satisfies $\widetilde{\mathbb{E}}[p(x)\cdot(P'(x_S^{(c)})-1)]=0$ for all multilinear polynomials $p(x)$ satisfying conditions 1 and 2. Let q be an arbitrary multilinear polynomial satisfying conditions 1 and 2. From (D.11), we see that

$$\sum_{f\in F}\widetilde{\mathbb{E}}[q(x)\cdot 1_{\{x_S=f\}}(x)\cdot(P'_f(x_S^{(c)})-1)]=0.$$

The result then follows from (D.13). ◀

E Correspondence between static LS$_+$roof system and relaxation

Recall the static LS$_+$ relaxation.

$$\sum_{\alpha\in\{0,1\}^S}p_S(\alpha)P(\alpha+c)\geq 1 \text{ for all } (c,S)\in\mathcal{I}$$
$$\{p_S\}_{S\subseteq[n],\,|S|\leq r} \text{ are } r\text{-locally consistent distributions} \tag{E.15}$$
$$\Sigma_{T=\alpha} \text{ is PSD } \forall T\subseteq[n],\alpha\in[q]^T \text{ such that } |T|\leq r-2, p_T(\alpha)>0.$$

A static LS$_+$ refutation has the following form.

$$\sum_\ell \gamma_\ell b_\ell(x)\phi_{I_\ell,J_\ell}(x)=-1, \tag{E.16}$$

where $\gamma_\ell\geq 0$, b_ℓ is an axiom or the square of an affine function, and $\phi_{I_\ell,J_\ell}=\prod_{i\in I_\ell}x_i\prod_{j\in J_\ell}(1-x_j)$.

▶ **Proposition 36.** *The static* LS$_+$ *SDP* (E.15) *is infeasible if and only if a static* LS$_+$ *refutation of the form* (E.16) *exists.*

Proof. Recall the definition of SA$_r$. We require that

$$\widetilde{\mathbb{E}}\left[\prod_{i\in I}x_i\prod_{j\in J}(1-x_j)\right]\geq 0 \tag{E.17}$$

for all $I,J\subseteq[n]$ such that $|I\cup J|\leq r$ and

$$\widetilde{\mathbb{E}}\left[a(x)\prod_{i\in I}x_i\prod_{j\in J}(1-x_j)\right]\geq 0 \tag{E.18}$$

for every axiom $a(x) \geq 0$ and for all $I, J \subseteq [n]$ such that $|I \cup J| \leq r$.

Using linearity of $\widetilde{\mathbb{E}}[\cdot]$, we can write this as a linear program in the variables $X_{I,J} := \prod_{i \in I} x_i \prod_{j \in J}(1 - x_j)$. Given a set $T \subseteq [n]$ and some assignment $\alpha : T \to \{0,1\}$, define α_0 to be $\{i \in T \ : \ \alpha(i) = 0\}$ and α_1 to be $\{i \in T \ : \ \alpha(i) = 1\}$. In (E.15), we additionally require that the matrices

$$M^{T,\alpha} = \left(\widetilde{\mathbb{E}}\left[x_i x_j \prod_{a \in \alpha_1} x_a \prod_{b \in \alpha_0} (1 - x_b)\right]\right)_{i \in [n], j \in [n]} = (X_{\alpha_1 \cup \{i,j\}, \alpha_0})_{i \in [n], j \in [n]} \tag{E.19}$$

are PSD for all $T \subseteq [n]$ such that $|T| \leq r$ and all $\alpha \in \{0,1\}^T$. As mentioned above, we can arrange the matrices $M^{T,\alpha}$ into a block diagonal matrix M such that M is PSD if and only if each of the $M^{T,\alpha}$'s are PSD. Let d be the dimension of M. Furthermore, we can think of the r-round SA constraints as being linear constraints on the entries of M. In particular, say these constraints have the form $A \cdot \mathrm{vec}(M) \geq b$, where $\mathrm{vec}(M) \in \mathbb{R}^{d^2}$ is the vector formed by concatenating the columns of M. Let c be the number of rows of A.

First, we show that the existence of a refutation of the form (E.16) implies that (E.15) is infeasible. Assume for a contradiction that there exists a solution $\{p_S\}_{S \subseteq [n],\, |S| \leq r}$ to (E.15). This implies the existence of a pseudoexpectation operator $\widetilde{\mathbb{E}}[\cdot]$ satisfying (E.17), (E.18), and (E.19). Now apply $\widetilde{\mathbb{E}}[\cdot]$ to each term of (E.16). The degree of each term $\gamma_\ell b_\ell(x) \phi_{I_\ell, J_\ell}(x)$ is at most r and we have that

$$\widetilde{\mathbb{E}}[\gamma_\ell b_\ell(x) \phi_{I_\ell, J_\ell}(x)] = \gamma_\ell \widetilde{\mathbb{E}}[b_\ell(x) \cdot \mathbf{1}_{\{x = \alpha\}}(x)],$$

where α is the unique assignment to $I_\ell \cup J_\ell$ such that $\phi_{I_\ell, J_\ell} = 1$. Let $U = \mathrm{supp}(b_\ell) \cup I_\ell \cup J_\ell$. If $b_\ell(x) \geq 0$ is an axiom, we know that $\widetilde{\mathbb{E}}[b_\ell(x) \cdot \mathbf{1}_{x=\alpha}(x)] \geq 0$ since every assignment β to U for which $p_U(\beta) > 0$ satisfies $b_\ell(x) \geq 0$. If b_ℓ is the square of some affine function, then PSDness of $\Sigma_{I_\ell \cup J_\ell = \alpha}$ implies that $\widetilde{\mathbb{E}}[\gamma_\ell b_\ell(x) \phi_{I_\ell, J_\ell}(x)] \geq 0$. Every term on the left hand side must be at nonnegative and we have a contradiction.

Now assume that (E.15) in infeasible. If consistent local distributions $\{p_S\}$ do not exist, then an SA refutation must exist and we are done. Assume, then, that consistent local distributions $\{p_S\}$ exist but the corresponding matrix M cannot be PSD. The sets $\{M \in \mathbb{R}^{d \times d} : A \cdot \mathrm{vec}(M) \geq b\}$ and $\{M \in \mathbb{R}^{d \times d} : M \text{ is PSD}\}$ are both nonempty, but their intersection is empty. We will need the following claim.

▶ **Claim 37.** *Let $S \subseteq \mathbb{R}^{d \times d}$ be convex, closed, and bounded. Suppose that for all $M \in S$, M is not PSD. Then there exists a PSD matrix $C \in \mathbb{R}^{d \times d}$ such that $C \cdot M < 0$ for all $M \in S$.*

Proof of Claim. The claim follows from the following two results.

▶ **Theorem 38** (Separating Hyperplane Theorem). *Let $S, T \subseteq \mathbb{R}^d$ be closed, convex sets such that $S \cap T = \emptyset$ and S is bounded. Then there exists $a \neq 0$ and b such that*

$a^\top x > b$ for all $x \in S$ and $a^\top x \leq b$ for all $x \in T$.

▶ **Lemma 39.** *A is PSD if and only if $A \cdot B \geq 0$ for all PSD B.*

Applied to our situation, the Separating Hyperplane Theorem says that there exists C and δ such that $C \cdot M < \delta$ for all $X \in S$ and $C \cdot M \geq \delta$ for all PSD X. We need to show that we can choose $\delta = 0$. Applying Lemma 39 will then complete the proof.

We know $\delta \leq 0$ because the zero matrix is PSD. It remains to show that we can choose $\delta \geq 0$. Assume for a contradiction that there exists PSD M such that $C \cdot M < 0$. We can then scale X by a large enough positive constant to get a PSD matrix M' such that $C \cdot M < \delta$, a contradiction. ◀

The claim implies that there is a PSD matrix C such that the set

$$\{M \in \mathbb{R}^{d\times d} : A \cdot \operatorname{vec}(M) \geq b\} \cap \{M \in \mathbb{R}^{d\times d} : C \cdot M \geq 0\}$$

is empty. As this set is defined by linear inequalities, we can apply Farkas' Lemma.

▶ **Theorem 40** (Farkas' Lemma). *Let $A \in \mathbb{R}^{m\times n}$ and consider a system of linear inequalities $Ax \geq b$. Exactly one of the following is true.*

1. *There is an $x \in \mathbb{R}^n$ such that $Ax \geq b$.*
2. *There is a $y \in \mathbb{R}^m$ such that $y \geq 0$, $y^\top A = 0$, and $y^\top b > 0$.*

In particular, this implies that there exist $y \in \mathbb{R}^c$ and $z \in \mathbb{R}$ such that

$$y^\top (A \cdot \operatorname{vec}(M) - b) + zC \cdot M < 0 \tag{E.20}$$

for all $M \in \mathbb{R}^{d\times d}$. Note that the first term is a nonnegative combination of SA constraints. Since C is PSD, we can write the eigendecomposition its $C = \sum_\ell \lambda_\ell v_\ell v_\ell^\top$ with $\lambda_\ell \geq 0$ for all ℓ. Also, recall that M is block diagonal with blocks $M^{T,\alpha}$. This block structure induces a corresponding partition of $[d]$. We can write the vector $v_\ell \in \mathbb{R}^d$ as $(v_{\ell,T,\alpha})_{T,\alpha}$ using this partition. Then the second term of (E.20) is

$$\begin{aligned} zC \cdot M &= z \sum_\ell \lambda_\ell (v_\ell v_\ell^\top) \cdot M \\ &= z \sum_\ell \lambda_\ell v_\ell^\top M v_\ell \\ &= z \sum_\ell \lambda_\ell \sum_{\substack{|T| \leq r-2 \\ \alpha \in \{0,1\}^T}} v_{\ell,T,\alpha}^\top M^{T,\alpha} v_{\ell,T,\alpha} \\ &= z \sum_\ell \lambda_\ell \sum_{\substack{|T| \leq r-2 \\ \alpha \in \{0,1\}^T}} \sum_{i,j \in [n]} v_{\ell,T,\alpha}(i) v_{\ell,T,\alpha}(j) M_{ij}^{T,\alpha}. \end{aligned}$$

Overall, we get

$$y^\top (A \cdot \operatorname{vec}(M) - b) + z \sum_\ell \lambda_\ell \sum_{\substack{|T| \leq r-2 \\ \alpha \in \{0,1\}^T}} \sum_{i,j \in [n]} v_{\ell,T,\alpha}(i) v_{\ell,T,\alpha}(j) M_{ij}^{T,\alpha} < 0.$$

Finally, we substitute in $X_{I,J} = \prod_{i\in I} x_i \prod_{j \in J}(1 - x_j)$ and scale appropriately to get an LS$_+$ refutation of the form (E.16). ◀

A No-Go Theorem for Derandomized Parallel Repetition: Beyond Feige-Kilian*

Dana Moshkovitz[1], **Govind Ramnarayan**[2], **and Henry Yuen**[†3]

1 **MIT, Cambridge, MA, USA**
`dmoshkov@mit.edu`

2 **MIT, Cambridge, MA, USA**
`govind@mit.edu`

3 **MIT, Cambridge, MA, USA**
`hyuen@mit.edu`

Abstract

In this work we show a barrier towards proving a randomness-efficient parallel repetition, a promising avenue for achieving many tight inapproximability results. Feige and Kilian (STOC'95) proved an impossibility result for randomness-efficient parallel repetition for two prover games with *small degree*, i.e., when each prover has only few possibilities for the question of the other prover. In recent years, there have been indications that randomness-efficient parallel repetition (also called *derandomized parallel repetition*) might be possible for games with large degree, circumventing the impossibility result of Feige and Kilian. In particular, Dinur and Meir (CCC'11) construct games with large degree whose repetition can be derandomized using a theorem of Impagliazzo, Kabanets and Wigderson (SICOMP'12). However, obtaining derandomized parallel repetition theorems that would yield optimal inapproximability results has remained elusive.

This paper presents an explanation for the current impasse in progress, by proving a limitation on derandomized parallel repetition. We formalize two properties which we call "fortification-friendliness" and "yields robust embeddings". We show that any proof of derandomized parallel repetition achieving almost-linear blow-up cannot both (a) be fortification-friendly and (b) yield robust embeddings. Unlike Feige and Kilian, we do not require the small degree assumption.

Given that virtually all existing proofs of parallel repetition, including the derandomized parallel repetition result of Dinur and Meir, share these two properties, our no-go theorem highlights a major barrier to achieving almost-linear derandomized parallel repetition.

1998 ACM Subject Classification F.2.3 Tradeoffs between Complexity Measures

Keywords and phrases Derandomization, parallel repetition, Feige-Killian, fortification

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.42

1 Introduction

1.1 Parallel Repetition and Almost Linear Blowup

Two prover games are central objects of study in probabilistically checkable proofs (PCPs) [1, 21, 16], cryptography [3, 4], and quantum computing [8, 22]. In a two prover game G, two all-powerful provers coordinate their strategies and are then sent to different rooms, where they can no longer communicate. A verifier samples a pair of correlated questions (x, y),

* This paper is based upon work supported by the National Science Foundation under grants number 1218547 and 1452302.

† Henry Yuen is additionally supported by Simons Foundation Fellowship (grant #360893).

and sends one question to each prover. Each prover sends back an answer, and the verifier accepts only if the pair of answers (a, b) satisfy some constraint $\pi_{(x,y)}$ depending on the questions. The *value* of the game G, denoted $\mathrm{val}(G)$, is the probability that the verifier accepts, maximized over all prover strategies.

Parallel repetition is a natural transformation to amplify the hardness of two prover games. The *k-fold parallel repetition* of a game G, denoted G^k, is another two prover game where the verifier picks k independent pairs of questions from G, and sends each prover k questions, corresponding to half of each of the k question pairs. Each prover sends back k answers and the verifier accepts if it would have accepted all k pairs of answers in the original game. Clearly, if the provers have strategies that make the verifier accept in the original game with probability 1 (i.e., $\mathrm{val}(G) = 1$), then they can make the verifier accept in the k-fold repetition with probability 1. The celebrated parallel repetition theorem of Raz [21] shows that if the value of the game G is smaller than 1, then the value of the k-fold repetition, $\mathrm{val}(G^k)$, decays exponentially with k.

One of the most important applications of parallel repetition is in hardness of approximation, where it is used in reductions proving inapproximability results [13]. However, this application reveals a significant disadvantage of parallel repetition: the randomness complexity of the verifier in G^k is k times the randomness complexity of the original game G. This increase corresponds to a blow-up of k in the exponent in reductions that are based on parallel repetition. As a result, if a reduction from SAT on size-n inputs applies k-fold parallel repetition to derive an instance of a target problem, then the resulting instance of the target problem takes inputs of size $O(n^k)$. Hence, the conjectured lower bound of $2^{\Omega(n)}$ on the time needed to solve SAT translates at best to a time lower bound of $2^{\Omega(n^{1/k})}$ on the target problem. In applications, k is often a large constant [13]. However, in order to obtain optimal inapproximability results for many problems, one would like to apply parallel repetition k times for all k's up to $\Theta(\log n)$ [2, 18].

This motivates the fundamental question of whether *derandomized* or *randomness efficient* parallel repetition is possible: could an analogue of Raz's parallel repetition theorem hold even if the verifier does not pick k question pairs independently, but rather picks k *correlated* question pairs? In particular, if the verifier of the original game uses $\log n$ random bits, one could hope for a verifier that uses $\log n + O(k)$ random bits to play the repeated game (as opposed to $k \log n$ random bits). If such a derandomized version of Raz's parallel repetition theorem were possible, then this would yield reductions from say, SAT, where a $2^{\Omega(n)}$ lower bound on SAT translates to a matching $2^{\widetilde{\Omega}(n)}$ lower bound on the target problem!

In [19], Moshkovitz and Raz gave a hardness amplification transformation similar in spirit to parallel repetition where the transformed game uses only $(1 + o(1)) \log n + O(k)$ random bits. Such a blowup is referred to as "almost linear", and is now the gold standard for reductions. Unfortunately, the answer size in the transformation of [19] is exponential in k rather than polynomial in k, and hence falls short of proving the so-called Projection Games Conjecture on optimal hardness of approximation. The parallel repetition transformation, on the other hand, gives an optimal tradeoff between the hardness of the resulting game (the *soundness error*) and the answer size. This motivates the search for a derandomized parallel repetition theorem that uses $(1 + o(1)) \log n + O(k)$ random bits and has $O(k)$ answer bits for all $k \leq \log n$. This could prove the Projection Games Conjecture, as well prove tighter inapproximability results.

1.2 The Feige-Kilian impossibility result

Feige and Kilian [12] proved an impossibility result for derandomized parallel repetition, showing that given a game G satisfying two conditions called *softness* and *small degree*, the value of any randomness-efficient parallel repetition of G is *independent* of the number of repetitions. The softness condition means that if G has randomness complexity $\log n$, then G is n^{ε}*-soft* iff on any subset of n^{ε} question pairs the verifier may ask, there exists a strategy for the provers to win with probability 1. A game has *degree-d* if for any question of one prover, the largest number of questions for the other prover is at most d. Specifically, their main result is the following:

▶ **Theorem 1** (Feige-Kilian). *Let G be a two prover game with n possible question pairs. If G is n^{ε}-soft and has degree d, then for any game H that involves playing k correlated instances of G, if the randomness complexity of the verifier of H is at most $c \log n$, then the value of H is independent of k; in particular,* $\mathrm{val}(H) \geq (2d)^{-4c^2/\varepsilon^2}$.

Here, we will call G the *base game* and H the *k-repeated game*. Next we describe the argument of Feige and Kilian in the almost linear regime (i.e., the repetition only uses $(1+\varepsilon)\log n$ random bits). In this regime their argument takes an especially simple form: because the base game has small degree, the provers have constant probability to guess each other's question in the first round, and if they succeed, there are only n^{ε} possibilities for the rest of the $k-1$ questions. For soft games the provers can succeed on all remaining questions – thus the provers' success probability in the repeated game does not decay with the number of repetitions k.

The softness condition is satisfied by games of interest. If we assume that solving Sat requires more than $2^{n^{\varepsilon}}$ time, then the games we apply parallel repetition to will be in general n^{ε}-soft. The small degree condition – while true of some games to which standard parallel repetition is applied – is not necessarily satisfied by all games of interest. In other words, Feige and Kilian's impossibility result imposes a strong limitation on the possibility of derandomized parallel repetition when working in the "small degree regime" – i.e., when the degree of G is a constant independent of the randomness complexity or the number of repetitions – but leaves the fascinating open question: can one obtain randomness-efficient parallel repetition for the "large degree regime", in which the degree of the game G can depend on its randomness complexity or the desired number of repetitions. In particular, Feige and Kilian do not rule out degree that is inversely proportional to the desired value of the repeated game.

Indeed, a few works have explored this avenue towards derandomized parallel repetition. Shaltiel [23] considered the setting of games where the questions to each prover are uncorrelated (also known as *free games*). Here, the degree is maximal, and Shaltiel managed to get a modest, albeit non-trivial, savings in randomness complexity in a repeated game. Dinur and Meir [10] constructed games with "linear structure" – which also have large degree – and showed that a theorem by Impagliazzo, Kabanets and Wigderson [15] gives a certain randomness-efficient parallel repetition for them. Unfortunately, neither of these results imply new hardness of approximation results, since the reductions from Sat to both free games and games with linear structure generate games with randomness complexity or answer size that are very large compared to the size of the Sat formula.

1.3 Our work

This paper begins where Feige and Kilian left off: we show a barrier for derandomized parallel repetition in the large degree regime. One may hope for an analogue of Feige and Kilian's negative result for large degree games, but, unfortunately, this seems to be impossible. The reason is that in fact there *are* games for which we can decrease error in a randomness efficient fashion, but without performing derandomized parallel repetition in a meaningful sense. Specifically, we can construct a high-error base game G that actually "hides" a game G_{low} for which we already know that $\text{val}(G_{low}) \leq \delta$; if then we apply a derandomized parallel repetition procedure such as Dinur's graph powering [9] to G, we obtain a repeated game H that closely approximates G_{low} and thus $\text{val}(H) \lesssim \delta \ll \text{val}(G)$. For more details, see Appendix C. Thus we've obtained derandomized error reduction, but intuitively the low error didn't come from the parallel repetition, but rather from the planted low-value game G_{low}.

This example shows that we can't hope to extend Theorem 1 directly to large degree games. Instead, we do the next best thing: we prove a limitation on *proof techniques* for derandomized parallel repetition. We formalize two proof properties which we call "fortification-friendliness" and "yields robust embeddings", and then show that any proof of almost-linear derandomized parallel repetition cannot simultaneously be fortification-friendly and yield robust embeddings. Nearly all proofs of parallel repetition – even derandomized parallel repetition theorems – are fortification friendly and yield robust embeddings, including: Raz [21], Shaltiel [23], Dinur-Meir [10], Impagliazzo, Kabanets and Wigderson [15], Moshkovitz [17], and Braverman-Garg [7]. Therefore our results explain why their techniques have not been pushed to almost linear size.

We now discuss these two properties in more detail.

1.3.1 Proof of parallel repetition by robust embedding

The key step in proofs of the parallel repetition theorem is to argue that the success probability of the average coordinate i of G^k cannot be much larger than $\text{val}(G)$, even when conditioned on the provers winning a significant fraction of coordinates that don't include i. This is proved via reduction: if this were not true, then the provers extract a strategy for G from a strategy for G^k by embedding G into the i'th coordinate of G^k conditioned on winning a set C of coordinates. However, if $\text{val}(G^n)$ is too large, then this strategy would succeed with probability better than $\text{val}(G)$, a contradiction. We say that such an analysis of parallel repetition is by *embedding.* Furthermore, the embeddings given are *robust.* By robust, we mean that embedding G into a coordinate of G^k is possible even when conditioning on winning *any* not too large subset C of coordinates. We will give a more detailed overview of this embedding technique in Section 3.

1.3.2 Fortification-friendly repetition schemes

Our no-go theorem covers derandomized parallel repetition theorems that can be applied to at least one *fortified* game. In this case we say that the parallel repetition theorem is *fortification-friendly.* Currently, there is no parallel repetition scheme that utilizes the fact that the base game is *not* fortified, and hence all existing parallel repetition schemes are fortification friendly. This includes the scheme of Dinur and Meir, which we elaborate on at the end of this subsection.

Fortification is a property of games introduced in [17]. Roughly speaking, a (δ, ε)-fortified game G is one where the value of so-called "rectangular" subgames of G that contain at least δ fraction of the questions is the same as the value of G up to an additive ε. The paper [17] gives a simple analysis for parallel repetition of fortified games, and furthermore showed that arbitrary games can be easily fortified by composing them with expanders.

While fortified games were defined fairly recently, they are quite natural, and, in particular, most games are fortified: see Appendix D.

Importantly, existing derandomized parallel repetition theorems are fortification-friendly: Shaltiel proves a derandomized parallel repetition for free games, which also works for fortified free games. In [10], Dinur and Meir first present a "linearization" operation that converts any game into a game with linear structure, and then prove a derandomized parallel repetition that works for *any* game with linear structure. The core of this derandomized parallel repetition is the work of Impagliazzo-Kabanets-Wigderson, and the underlying derandomized parallel repetition theorem of [15] *is* fortification friendly. This is because the result of Impagliazzo-Kabanets-Wigderson applies to all free games: (1) free games trivially have linear structure (since all possible edges are present) and (2) it is easy to construct fortified free games (e.g. choosing random constraints for a free game). Thus, our results imply limitations on what is achievable by the Impagliazzo-Kabanets-Wigderson derandomized parallel repetition, and hence what is achievable by the Dinur-Meir result.

1.3.3 Informal Theorem Statement and Discussion

We are now ready to state our main theorem informally. For a formal statement, see Theorem 4.

▶ **Theorem 2** (Main theorem, informal statement). *Let $\mathcal{S}$ be a parallel repetition scheme that transforms any* base game *G to a k-repeated game $\mathcal{S}(G)$ in which a verifier asks k (possibly correlated) questions from G in parallel. Suppose that G is (δ, ε)-fortified for sufficiently small*[1] *$\delta = |G|^{-o(1)}$, and $\varepsilon = O(1 - \mathrm{val}(G))$; and that $|\mathcal{S}(G)| = |G|^{1+o(1)}$. Then there is no proof of parallel repetition by robust embedding for $\mathcal{S}(G)$.*

Note that unlike the result of Feige and Kilian, our impossibility result is not limited to small degree games. In fact, fortification typically involves composing the game with a degree-$O(1/\delta)$ expander, thereby making the degree of the base game large.

1.4 The way forward

Despite many years of research on the subject of derandomized parallel repetition, obtaining a parallel repetition with both an exponential decay of the error and almost-linear blowup has resisted attack. The work of Dinur and Meir makes partial progress towards this goal, but – not only it admits polynomial decay of the error and a large polynomial blowup – it also goes through a costly "linearization" operation that deteriorates the parameters of the game, so it does not achieve any new results for PCP.

We view our theorem as an explanation for the lack of progress towards the goal of derandomized parallel repetition. It shows that any proof of a derandomized parallel repetition theorem must do at least one of the following: (1) Use that the base game is *not* fortified; (2) Not yield a robust embedding; and/or (3) Have a large polynomial blowup. As discussed earlier, virtually all proofs of parallel repetition do not satisfy (1) and (2). We now discuss prospects for being able to achieve (1), (2), or (3).

[1] The required δ depends on the blowup in $\mathcal{S}(G)$. For $|\mathcal{S}(G)| = |G| \cdot \mathrm{poly}\log|G|$, we need $\delta = 1/\mathrm{poly}\log|G|$. For $|\mathcal{S}(G)| = O(|G|)$, we need a sufficiently small constant δ.

Using that the base game is *not* fortified

Is it too restricting to require that the scheme accepts a base game is fortified? We believe not, there are no known parallel repetition techniques that take advantage of the base game not being fortified. Intuitively, it seems unlikely that such a technique would help with derandomized parallel repetition, since fortification is known to facilitate parallel repetition, and composition with expanders is intuitively useful for derandomization.

Circumventing robust embeddings

Again, proving parallel repetition via robust embeddings (either explicitly or implicitly) is a ubiquitous strategy. Interestingly, one approach that does not fall into the embedding framework is the randomness-efficient amplification of Moshkovitz and Raz [19]. They construct codes with local testers/decoders *that have low error*, and incorporate randomness efficient sequential repetition on the decoded symbols. Their technique is based on an algebraic construction of codes and the error it obtains, while low, is not low enough to prove the Projection Games Conjecture. Decreasing the error of local testers/decoders does not seem any easier than randomness-efficient error reduction for games.

Polynomial blowup

Finally, our impossibility result pertains to repetitions with almost linear blowup. As we mentioned, such a blowup is currently the gold standard in PCP, and larger blowups correspond to weaker inapproximability results. Nonetheless, both the results of Shaltiel and Dinur-Meir have larger blowups. Shaltiel has a blowup that is not much smaller than standard parallel repetition, and Dinur-Meir have a polynomial blowup.

2 Games and parallel repetition schemes

We will use the notation $\overline{x}$ to denote tuples $(x_1, \ldots, x_k)$. For convenience of notation, we will call two sets ε-close if the uniform distributions on these sets are ε-close in total variation distance.

Games and strategies

A *two-prover one-round game* G is specified by a tuple (X, Y, E, π, Σ) where $X \times Y$ is the vertex set of a bipartite graph with edge set $E \subseteq X \times Y$, π is a set of constraints $\pi_e \subseteq \Sigma \times \Sigma$ for each edge $e \in E$, and Σ is a finite alphabet. The *value* of a game G is defined as

$$\mathrm{val}(G) := \max_{\psi_X, \psi_Y} \Pr_{(x,y)\in E}\left[(\psi_X(x), \psi_Y(y)) \in \pi_{(x,y)}\right]$$

where the maximum is taken over all functions $\psi_X : X \to \Sigma$ and $\psi_Y : Y \to \Sigma$, and the probability is over a uniformly random edge in E. We will use caligraphic $\mathcal{G}$ to denote the *graph underlying* G, which is the bipartite graph (X, Y, E). The *size* of a game G, which we will denote by $|G|$, is defined to be the number of edges $|E|$. For a pair of maps $\psi_X : X \to \Sigma$ and $\psi_Y : Y \to \Sigma$, we call $\psi = (\psi_X, \psi_Y)$ a *strategy* for G. For $(x, y) \in X \times Y$, we will write $\psi(x, y)$ to denote the pair $(\psi_X(x), \psi_Y(y))$. If the maximum degree of $\mathcal{G}$ is d, then we say that G is a *degree-d game*.

k-fold parallel repetition

The *k-fold parallel repetition* of a game G is a new game $G^k = (X^k, Y^k, E^k, \pi^k, \Sigma^k)$, where X^k, Y^k, E^k, and Σ^k denote the k-fold Cartesian products of X, Y, E and Σ respectively, and π^k denotes the set of constraints $\pi_{e_1} \times \pi_{e_2} \times \cdots \times \pi_{e_k}$ for every $\overline{e} = (e_1, \ldots, e_k) \in E^k$. Intuitively, in G^k, the verifier will sample $e_1 = (x_1, y_1), \ldots, e_k = (x_k, y_k)$ uniformly and independently at random from E, and send $\overline{x} = (x_1, \ldots, x_k)$ and $\overline{y} = (y_1, \ldots, y_k)$ to the first and second prover, respectively. The provers win the repeated game G^k if they win G in all rounds – i.e., provide answers $(a_1, \ldots, a_k)$ and $(b_1, \ldots, b_k)$ from Σ^k such that for $i \in [k]$, $(a_i, b_i) \in \pi_{e_i}$.

Subgames

Let $G = (X, Y, E, \pi, \Sigma)$ be a game. Then we say a game $G' = (X', Y', E', \pi', \Sigma')$ is a *subgame of G* if $X' \subseteq X$, $Y' \subseteq Y$, $E' \subseteq E \cap (X' \times Y')$, $\pi' = \{\pi_e : e \in E', \pi_e \in \pi\}$, and $\Sigma' = \Sigma$; we denote this by $G' \subseteq G$. For a subset $E' \subseteq E$, we will let $G_{E'} = (X, Y, E', \pi, \Sigma)$ denote the *subgame of G induced by E'*. Notice that the question set, constraints and alphabet of a subgame induced by a set of edges are the same as that of the original game. The only difference is that, in the subgame, we only select a subset of the question pairs that the verifier can ask, and the constraints are induced by the subset of questions.

For convenience, when the game $G = (X, Y, E, \pi, \Sigma)$ is understood from context, we will treat G as the set of edges E; e.g., we will write $(x, y) \in G$ to denote $(x, y) \in E$.

Parallel repetition schemes

Let $G = (X, Y, E, \pi, \Sigma)$ be a game, and let $k > 0$ be an integer. Then we say any subgame $H = (X^k, Y^k, E_H, \pi^k, \Sigma^k) \subseteq G^k$ where $E_H \subseteq E^k$ is a *k-repeated game*, with G as the *base game*. If $|H|$ is strictly smaller than $|G|^k$, then we say that H is a *derandomized* k-repeated game.

A *k-parallel repetition scheme* $\mathcal{S}$ is a black box procedure for converting a base game G to a k-repeated game $H \subseteq G^k$. In this paper, we will use the shorthand $\mathcal{S} = \{G \to H \subseteq G^k\}$ to succinctly describe the scheme $\mathcal{S}$, where we implicitly assume the transformation $G \to H$ is described by an algorithm that runs in time polynomial in the description of the input game, as well as k. Whenever a parallel repetition scheme (or simply a *repetition scheme*) $\mathcal{S}$ is understood from context, H will always refer to the k-repeated game that is the scheme $\mathcal{S}$ applied to some base game G. We will also use $\mathcal{S}(G)$ to denote the scheme applied to G.

We say that a k-parallel repetition scheme $\mathcal{S} = \{G \to H \subseteq G^k\}$ satisfies the *uniform marginals property* if for all games $G = (X, Y, E, \pi, \Sigma)$, the marginal distribution of questions sampled from $H = (X^k, Y^k, E_H, \pi^k, \Sigma^k) = \mathcal{S}(G)$ in any single coordinate is the same as the distribution of questions in G. Namely, for all coordinates $j \in [k]$ and any fixed edge $(x, y) \in E$, we have that

$$\Pr_{(\overline{x},\overline{y}) \in E_H} \left[(\overline{x}_j, \overline{y}_j) = (x, y) \right] = \frac{1}{|E|}.$$

The uniform marginals property is an extremely mild and natural condition, which holds for all existing parallel repetition schemes. In fact, this condition even seems morally necessary for parallel repetition, as it says that each coordinate of the repeated game H should look like the base game G, which is what we expect of a repeated game.

Finally, we define the *size blowup* of a scheme $\mathcal{S}$ to be a function $\Phi_{\mathcal{S},\Sigma} : \mathbb{N} \to \mathbb{R}$ defined as

$$\Phi_{\mathcal{S},\Sigma}(n) := \max_{G:|G|=n} \frac{|H|}{|G|}$$

where the maximum is over all base games G with n question pairs and answer alphabet Σ, and H denotes the scheme $\mathcal{S}$ applied to G. Note that the number of games with n question pairs and answer alphabet Σ is finite[2]. The size blowup of a scheme captures the blowup in randomness complexity in the following way: if the base game G has randomness complexity $\log n$ and the k-repeated game H has randomness complexity at most $\log n + \ell(n)$, then the size blowup $\Phi_{\mathcal{S},\Sigma}(n) \leq 2^{\ell(n)}$.

Winning in a set of coordinates

For any k-repeated game H, any strategy ψ for H, and any subset of coordinates $C \subseteq [k]$, let W_C^ψ denote the subgame of H consisting of all question pairs $(\overline{x}, \overline{y}) \in H$ such that $(\overline{a}, \overline{b}) = \psi(\overline{x}, \overline{y})$ satisfies $(\overline{a}_i, \overline{b}_i) \in \pi_{\overline{x}_i, \overline{y}_i}$ for all $i \in C$. In other words,W_C^ψ is the set of all question pairs in H where the strategy ψ is able to succeed in all the coordinates of C. We call W_C^ψ the *subgame where ψ wins in C*. When the strategy is ψ is understood from context, we will omit ψ and simply write the subgame as W_C.

3 Parallel repetition via embeddings

In this section, we formalize the notion of an embedding as described in the introduction and expand on how it is used to prove parallel repetition. First, we will motivate the idea of embedding by giving a high level and informal discussion of proofs of parallel repetition. Then, we will formally define the notion of a *robust embedding* that we will use in this paper. The idea of robust embeddings is implicit in nearly all proofs of parallel repetition: to illustrate this, we show how it is implicit in the Raz-Holenstein proof in Appendix A.

As alluded to in the introduction, most proofs of parallel repetition proceed via *reduction*: the value of the repeated game G^k is related to the value of the base game G by exhibiting a transformation that takes a "too good" strategy for the repeated game G^k and constructs a "too good" strategy for the base game G. Furthermore, this transformation is black box, in the sense that it works for arbitrary games G and their parallel repetitions.

How might such a generic transformation work? Intuitively, it seems that one must have a generic way of identifying a substructure within a hypothetical too-good-to-be-true strategy ψ for the repeated game G^k, a strategy φ for the base game G that succeeds with too-high probability (i.e., strictly greater than $\text{val}(G)$, which would be a contradiction). Since our only constraint on G^k is that it's comprised of k independent copies of G, it seems that we have to identify a strategy for G within substructures of ψ that respect this constraint.

Under ψ, we have that $\Pr[W_{[k]}]$, the probability of winning all rounds, is too large. Thus, we can use Bayes' rule to split it into conditional probabilities that respect the coordinate structure of G^k. It is not hard to see that, assuming $\Pr[W_{[k]}]$ is too large, then there exists a set of coordinates $C \subset [k]$ such that for many rounds $i \in [k] \backslash C$, we have that $\Pr[W_{\{i\}} | W_C] \gg \text{val}(G)$, where $W_{\{i\}}$ denotes the event of winning round i and W_C denotes

[2] We assume that in the transformation from base game G to k-repeated game H, the scheme does not care about the actual labels of the questions, and what only matters are the correlations between questions, as captured by the edge set E of the base game G. This is consistent with existing parallel repetition schemes.

the event of winning all the rounds in C. Thus for each such i it appears that we have identified candidate substructures inside ψ (namely, the event W_C) within which we hope to extract a too-good-to-be-true strategy for G (namely, by using a strategy for the ith round within the event W_C). Thus, we would like to "play" a copy of G in the ith round of W_C, and obtain success probability that is close to $\Pr[W_{\{i\}}|W_C]$, which would be too good to be true. The constructed strategy φ will attempt to "play", or *embed*, the questions of G into the i'th round of W_C (which we also think of as a subgame of G^k).

We call this natural proof strategy a *proof of parallel repetition by robust embedding.* This proof strategy forms the basis of most parallel repetition proofs, including existing proofs of derandomized parallel repetition, and one might expect that future derandomized parallel repetition theorems might be proved along these lines. We formalize this notion by defining the property of having a *robust embedding* of a game G into a k-repeated game H.

Let $G = (X_G, Y_G, E_G, \pi_G, \Sigma_G)$ and $H = (X_H, Y_H, E_H, \pi_H, \Sigma_H)$ be games. We say the map $\mathsf{Emb} : X_G \times Y_G \to X_H \times Y_H$ is an *embedding map from G to H* (or simply an *embedding map*) iff there exist maps $f : X_G \to X_H$ and $g : Y_G \to Y_H$ such that for all $(x, y) \in X_G \times Y_G$ we have $\mathsf{Emb}(x, y) = (f(x), g(y))$.

▶ **Definition 3** (Robust embedding into a repeated game)**.** Let $G = (X, Y, E, \pi, \Sigma)$ be a game and let $H \subseteq G^k$ be a k-repeated game. Let $\gamma : [k] \to \mathbb{R}$ be a function. We say that G has a (γ, ε)-*robust embedding into a coordinate of H* iff for all strategies ψ_H for H and subsets $C \subseteq [k]$, there exists an $i \in [k]\backslash C$, there exists an embedding map $\mathsf{Emb} : X \times Y \to X^k \times Y^k$ such that

1. **(Coordinate embedding)** For all $(x, y) \in X \times Y$, we have that $(\overline{x}, \overline{y}) = \mathsf{Emb}(x, y)$ satisfies $\overline{x}_i = x$ and $\overline{y}_i = y$.
2. **(Robustness)** If $\Pr_{\overline{e} \in E_H}[\overline{e} \in W_C] \geq \gamma(|C|)$, then $\Pr_{e \in E}[\mathsf{Emb}(e) \in W_C] \geq 1 - \varepsilon$.

where E_H denotes the questions in H, and W_C denotes the subgame of H where ψ_H wins in C.

We use the term "robust" because there is an embedding from G into W_C for *every* C such that $\Pr[W_C]$ is sufficiently large. This is reminiscent of the robustness properties of pseudorandom objects such as expanders or extractors, where we have guarantees for *every* sufficiently large subset of a graph (in the case of expanders) or distribution with sufficiently large min-entropy (in the case of extractors). *A priori*, G may not have a robust embedding into a repeated game $H \subseteq G^k$ because there may exist large $W_C \subseteq H$ that, intuitively, does not contain a copy of G.

Our definition of robust embedding is heavily inspired by the Raz-Holenstein proof of the parallel repetition theorem. In Appendix B, we explicitly describe how the Raz-Holenstein proof directly implies the existence of a robust embedding of G into a coordinate of G^k.

Although the main result of our paper does not unconditionally rule out derandomized parallel repetition, we do the next best thing: we rule out a particular *proof technique* for proving derandomized parallel repetition, and in fact, a very natural one.

4 Our no-go theorem

Our main theorem is the following:

▶ **Theorem 4** (Main Theorem)**.** *Let Σ be a finite alphabet. Let $\mathcal{S} = \{G \to H \subseteq G^k\}$ be a parallel repetition scheme that satisfies the uniform marginals property and has size blowup $\Phi_{\mathcal{S},\Sigma}(n) \leq O(n^{0.49})$. Then for all $n > 0$, $\varepsilon \in (0, 1/23)$, $\delta \leq (16\Phi_{\mathcal{S},\Sigma}(n) \log^2(\Phi_{\mathcal{S},\Sigma}(n)))^{-1}$, an integer d, and for all games G satisfying:*

1. *The graph $\mathcal{G} = (X \times Y, E)$ underlying G is d-regular, and has at most $\varepsilon|E|$ parallel edges.*
2. *For all $S \subseteq X, T \subseteq Y$ with $|S| \geq \delta|X|, |T| \geq \delta|Y|$, we have*

$$\left|\frac{|E \cap (S \times T)|}{|S||T|} - \frac{d}{|Y|}\right| \leq \varepsilon\frac{d}{|Y|}.$$

3. $\mathrm{val}(G) \leq 1 - 20\varepsilon$.
4. *G is (δ, ε)-fortified.*

there does not exist a (γ, ε)-robust embedding of G into a coordinate of H for all $\gamma \leq \mathrm{val}(G)$, $\varepsilon < (1 - \gamma)/23$, where $H = \mathcal{S}(G)$.

The most significant implication of our Main Theorem is that a parallel repetition scheme satisfying the uniform marginals property that (1) can be applied to a single game G with the above properties and (2) yields a robust embedding cannot achieve almost linear blowup. As we elaborate below, by far the most pertinent property of G is that it is sufficiently *fortified*. Hence, a parallel repetition scheme attempting to achieve almost linear blowup should explicitly take advantage of the fact that its input is not fortified. Below, we discuss the properties we require of G, and why games that satisfy these properties are quite natural, which makes this barrier nontrivial to overcome.

Fortification is a property of games introduced by Moshkovitz [17], who gave a simple proof that fortified games satisfy parallel repetition, and furthermore showed that arbitrary (projection) games can be easily fortified. Roughly speaking, a fortified game G is one where the value of every not-too-small *rectangular subgame* of G (i.e., a subgame of G played on a subgraph of $\mathcal{G}$ induced by a set of vertices). More formally:

▶ **Definition 5** (Fortified Games)**.** Let $G = (X, Y, E, \pi, \Sigma)$ be a game. We say that G is (δ, ε)-*fortified* iff for all $S \subseteq X$, $T \subseteq Y$ with $|S| \geq \delta|X|$ and $|T| \geq \delta|Y|$, we have that

$$\mathrm{val}(G_{S\times T}) \leq \mathrm{val}(G) + \varepsilon$$

where $G_{S\times T} \subseteq G$ denotes the rectangular subgame of G on the subgraph induced by the vertex set $S \times T$.

One might be cautious that we require G to be (δ, ε)-fortified with potentially polynomially small δ. However, many natural games on which parallel repetition theorems apply *are* this fortified. For example, we demonstrate in Appendix D that games that are randomly sampled from a distribution of games on regular bipartite graphs are heavily fortified. There are even heavily fortified games with linear structure: it follows from Claim 19 in Appendix D that by taking a free game and randomly sampling constraints, we get a game that satisfies all the requirements of Theorem 4. Since all free games have linear structure, this also has linear structure. Hence, our barrier even applies to the derandomized parallel repetition theorem of Dinur and Meir.

Another property that we require of the game G is that it is a d-regular, bipartite expander. However, this is not restrictive as an additional property: all known constructions of fortified games satisfy the expansion condition we desire. This includes those in [17] and [5], which create fortified games by composing games with good expanders, as well as the random games we construct in Appendix D, which are naturally on expanders. The last property that we use is that the graph underlying G does not have too many parallel edges, which is a property satisfied by virtually all games researchers consider in hardness of approximation. Finally, the reason we limit the blowup of the parallel repetition scheme $\mathcal{S}$ to be at most $O(n^{0.49})$ is because we take δ to be inversely proportional to the blowup.

Since n is the number of *edges* in the underlying graph, taking $\delta = O(n^{-0.5})$ already makes $\delta|X|$ and $\delta|Y|$ smaller than 1 for free games, and hence (δ, ε)-fortification simply does not make sense. Furthermore, we believe the most interesting case of our theorem occurs when the blowup is just $O(n^{o(1)})$, in which case our fortification constraints are relatively mild.

We now give an intuitive overview of our argument. Let $\mathcal{S}$ be the parallel repetition scheme from the theorem statement satisfying the uniform marginals property and the small size blowup condition. We will show that for games G satisfying the requisite properties, the presence of a robust embedding lets us obtain a contradiction.

Given such a game G and a supposed randomness-efficient parallel repetition $H \subseteq G^k$ from the scheme $\mathcal{S}$, we rule out the existence of a robust embedding of G into H. We prove this via contradiction: if there were a robust embedding, then from the embedding we would be able to extract an assignment for G that has success probability significantly greater than $\mathrm{val}(G)$ on some rectangular subgame of G. Furthermore, the fact that H is not much larger than G allows us to conclude that this rectangular subgame is not too small. However, this contradicts the fortification property of G, which states that all not-too-small rectangular subgames of G have value that's not much larger than $\mathrm{val}(G)$. Thus no such robust embedding can exist.

We give more details about how we extract an assignment from a robust embedding. Recall that a robust embedding of G to H allows us to choose a subset of coordinates $C \subseteq [k]$ and a strategy ψ^H for the repeated game such that, if under ψ^H the probability of success in the coordinates C is greater than some threshold γ (which depends on the size of C), then there exists an embedding Emb that maps G into the subgame W_C of H where the provers win in C.

We exploit this by letting C be a singleton round $\{s\}$, and letting ψ^H be a trivial strategy where the provers play optimally in round s, and all other rounds independently.[3] The probability of succeeding in round s under this strategy is precisely $\mathrm{val}(G)$, which is larger than the threshold γ. Therefore we obtain an embedding Emb from G into the subgame $W_{\{s\}} \subseteq H$ where the provers win in round s.

Then, we use the fact that H is a randomness-efficient parallel repetition of G and the uniform marginals property to conclude that over the question pairs (x, y) in the base game G, the projection of $\mathsf{Emb}(x, y)$ (which are question pairs in the repeated game H) onto round s must contain a rectangular subgame $G_{M\times N}$ of G that has substantial size. Since G is (approximately) embedded into $W_{\{s\}}$, by definition, ψ^H yields an (almost-)satisfying assignment for $G_{M\times N}$. As stated previously, this would violate the fortification property of G.

Though the intuition is rather straightforward, much of the proof involves dealing with the fact that G doesn't *perfectly* embed into $W_{\{s\}}$, but only approximately so, which introduces errors in extracting a nearly satisfying assignment for a rectangular subgame of G. We defer the full proof of Theorem 4 to Appendix A.

[3] One might find it suspicious that we're deriving a robust embedding from such a trivial strategy, whereas in the proof of Raz's parallel repetition, for example, a robust embedding is derived from "too-good-to-be-true" strategies. However, one can see from Appendix B that the Raz-Holenstein proof of parallel repetition does indeed give us a robust embedding into the subgame $W_{\{s\}}$ from this trivial strategy. Furthermore, as described in Section 3 and Appendix B, a robust embedding is necessary but not sufficient for proving parallel repetition, which is why the robust embedding derived from the trivial strategy won't contradict the fact that the success probability for this strategy is less than $\mathrm{val}(G)^k$.

5 Conclusion and Open Problems

We show limitations on a prevalent proof strategy for derandomized parallel repetition. Specifically, we prove that any parallel repetition scheme that can be applied to a fortified game and yields a "robust embedding" cannot achieve almost-linear blowup. We leave it as an open problem to extend our limitation to schemes with larger blowup. An intriguing related question is whether one can extend our results to provide limitations on derandomized parallel repetition schemes with polynomial blow-up and exponential soundness decay. This would not contradict existing results: Shaltiel's repetition has exponential soundness decay but has nearly-exponential blowup, and Dinur-Meir achieve polynomial blow-up but have polynomial soundness decay. As we discussed in the Introduction, the limitation of Feige-Kilian is simple in the case of almost-linear blowup, whereas the case of large blowup is considerably more complicated, and it is possible that extending our result to large polynomial blowup will be similarly difficult.

Our analysis takes a robust embedding and extracts from it fairly large rectangles that are nearly satisfied. The limitation follows from providing fortified games, which do not have such rectangles, as input. An intriguing possibility given this state of affairs is the following: Is there a technique for parallel repetition that explicitly makes use of lack of fortification in the input? Such a technique would be able to circumvent our limitation if it were applicable to derandomized parallel repetition.

A direction for amplifying two prover games that is not captured by our limitation is amplification via locally decode or reject codes [19]. These are efficient encodings with a two query tester/decoder. The tester/decoder is able to decode k-tuples of symbols from its message, or identify a corruption in the word. One can encode the answers of the players via such a code, and then ask each prover a different query of the tester/decoder. Whenever the tester/decoder is correct, one can simulate a randomness-efficient sequential repetition of the base game. There are constructions of locally decode or reject codes based on low degree polynomials (See [19] and many previous works), or based on direct product testing (See [15, 10] and many previous works). The value of the amplified game is typically inherited from the error probability of the local tester/decoder. It remains an open problem to find locally decode or reject codes with substantially lower error than existing constructions.

Acknowledgments. We thank Pritish Kamath and Irit Dinur for helpful discussions. We also thank the anonymous reviewers for their suggestions on the initial manuscript.

References

1 L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 16–25. IEEE, 1990.

2 M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximations. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 294–304, 1993.

3 M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 113–131. ACM, 1988.

4 M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Efficient identification schemes using two prover interactive proofs. In *Advances in Cryptology – CRYPTO'89 Proceedings*, pages 498–506. Springer, 1990.

5 A. Bhangale, R. Saptharishi, G. Varma, and R. Venkat. On fortification of projection games. *arXiv*, 2015. URL: `http://arxiv.org/abs/1504.05556`.

6 Andrej Bogdanov. Gap amplification fails below 1/2, 2005. Comment on ECCC TR05-046, can be found at `http://eccc.uni-trier.de/eccc-reports/2005/TR05-046/commt01.pdf`.

7 M. Braverman and A. Garg. Small value parallel repetition for general games. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 335–340. ACM, 2015.

8 R. Cleve, P. Høyer, B. Toner, and J. Watrous. Consequences and limits of nonlocal strategies. In *Computational Complexity, 2004. Proceedings. 19th IEEE Annual Conference on*, pages 236–249. IEEE, 2004.

9 I. Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

10 I. Dinur and O. Meir. Derandomized parallel repetition via structured PCPs. *Computational Complexity*, 20(2):207–327, 2011.

11 I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 624–633. ACM, 2014.

12 U. Feige and J. Kilian. Impossibility results for recycling random bits in two-prover proof systems. In *Proc. 27th ACM Symp. on Theory of Computing*, pages 457–468, 1995.

13 J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

14 T. Holenstein. Parallel repetition: Simplification and the no-signaling case. *Theory of Computing*, 5(1):141–172, 2009.

15 R. Impagliazzo, V. Kabanets, and A. Wigderson. New direct-product testers and 2-query PCPs. *SIAM Journal on Computing*, 41(6):1722–1768, 2012.

16 S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775. ACM, 2002.

17 D. Moshkovitz. Parallel repetition from fortification. In *Proc. 55th IEEE Symp. on Foundations of Computer Science*, pages 414–423, 2014.

18 D. Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. *Theory of Computing*, 11(7):221–235, 2015.

19 D. Moshkovitz and R. Raz. Two query PCP with sub-constant error. *Journal of the ACM*, 57(5), 2010.

20 A. Rao. Parallel repetition in projection games and a concentration bound. *SIAM Journal on Computing*, 40(6):1871–1891, 2011.

21 R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27:763–803, 1998.

22 B.W. Reichardt, F. Unger, and U. Vazirani. Classical command of quantum systems. *Nature*, 496(7446):456–460, 2013.

23 R. Shaltiel. Derandomized parallel repetition theorems for free games. *Computational Complexity*, 22(3):565–594, 2013.

A Proof of Theorem 4 (Main Theorem)

In this section, we prove the main theorem of this paper, which we restate here for clarity:

▶ **Theorem 6** (Main Theorem). *Let Σ be a finite alphabet. Let $\mathcal{S} = \{G \to H \subseteq G^k\}$ be a parallel repetition scheme that satisfies the uniform marginals property and has size blowup $\Phi_{\mathcal{S},\Sigma}(n) \le O(n^{0.49})$. Then for all $n > 0$, $\varepsilon \in (0, 1/23)$, $\delta \le (16\Phi_{\mathcal{S},\Sigma}(n)\log^2(\Phi_{\mathcal{S},\Sigma}(n)))^{-1}$, an integer d, and for all games G satisfying:*

1. *The graph $\mathcal{G} = (X \times Y, E)$ underlying G is d-regular, and has at most $\varepsilon|E|$ parallel edges.*

2. *For all $S \subseteq X, T \subseteq Y$ with $|S| \geq \delta|X|, |T| \geq \delta|Y|$, we have*

$$\left| \frac{|E \cap (S \times T)|}{|S||T|} - \frac{d}{|Y|} \right| \leq \varepsilon \frac{d}{|Y|}.$$

3. $\mathrm{val}(G) \leq 1 - 20\varepsilon$.
4. *G is (δ, ε)-fortified.*

there does not exist a (γ, ε)-robust embedding of G into a coordinate of H for all $\gamma \leq \mathrm{val}(G)$, $\varepsilon < (1-\gamma)/23$, where $H = \mathcal{S}(G)$.

Let $\mathcal{S}$, n, Σ, ε, δ, and d be as in the theorem statement. In the proof we let d_X and d_Y denote the left-degree and right-degree of $\mathcal{G}$, and so $d_X = d_Y = d$, since $\mathcal{G}$ is d-regular.

Let $H = (X^k, Y^k, E_H, \pi^k, \Sigma^k)$ be the k-repetition of G under the scheme $\mathcal{S}$. Define $z := \frac{|H|}{|G|}$. Notice that $z \leq \Phi_{\mathcal{S},\Sigma}(n)$, since $\Phi_{\mathcal{S},\Sigma}(n)$ is effectively a maximum of z taken over all games G with $|G| = n$. Fix $\gamma \leq \mathrm{val}(G) \leq 1 - 20\varepsilon$. Fix a round $s \in [k]$. Let $\psi^H = (\psi^H_X, \psi^H_Y)$ be a strategy for the provers in the repeated game under which the event of winning in round s occurs with probability at least γ. Note that such a strategy always exists: the two provers can play the optimal strategy for G in round s, and by the uniform marginals property of Theorem 4, $\Pr[W_{\{s\}}] = \mathrm{val}(G) \geq \gamma$.

Suppose for contradiction that there exists an (γ, ε)-robust embedding into a coordinate of H. Let $C = \{s\}$ for some round $s \in [k]$. Then by definition of a robust embedding, we obtain an embedding map $\mathsf{Emb}(x, y) = (f_X(x), f_Y(y))$ for maps $f_X : X \to X_H$ and $f_Y : Y \to Y_H$. Denote by $W_{\{s\}}$ the set of edges that win in round s under the strategy ψ^H.

Define W to be the set of edges that win in round s and are mapped into by the embedding map, namely, $W := \{\mathsf{Emb}(x, y) : (x, y) \in E\} \cap W_{\{s\}}$, and let H_W be the subgame of H induced by the edge set W.

Combining the fact that Emb is a robust embedding into $\{s\}$ (Definition 3, Property 2) with the definition of W, we know that

$$\Pr_{(x,y)\in E}[\mathsf{Emb}(x, y) \in W] \geq 1 - \varepsilon \tag{1}$$

While all but an ε-fraction of edges $(x, y) \in E$ map into W under Emb, it will be convenient for us to define a set $\widehat{W}$ which *all* the edges (x, y) map into. Hence, we define the set of repeated game vertex pairs $\widehat{W}$ to be

$$\widehat{W} = \{\mathsf{Emb}(x, y) : (x, y) \in E\} \subseteq X^k \times Y^k .$$

By equation (1), we observe that

$$|\widehat{W} \backslash W| \leq \varepsilon |\widehat{W}| = \varepsilon |E|$$

and hence, W and $\widehat{W}$ are ε-close. Note that some of the edges in $\widehat{W}$ may not exist in E_H. However, it will be useful to think of $\widehat{W}$ as a set of edges that induces a graph on repeated game vertices $\mathcal{H}_{\widehat{W}} = (im(f_X), im(f_Y), \widehat{W})$. While we use the notation $\mathcal{H}_{\widehat{W}}$ to indicate that it is a graph on repeated game vertices, it is again important to note that this graph is not a subgraph of $\mathcal{H}$.

For the remainder of the proof, we will assume that W and $\widehat{W}$ have no parallel edges, and that $\mathcal{H}_{\widehat{W}}$ is isomorphic to G. We note that this is not strictly true if G has parallel edges - however, since we know that G has at most $\varepsilon|E|$ parallel edges so $\mathcal{H}_{\widehat{W}}$ is ε-close to a graph that is isomorphic to G even after taking out parallel edges. Hence, the same argument goes through by simply making ε slightly smaller. For a more detailed discussion of how we handle a small number of parallel edges, we refer the reader to the Remark at the end of this section.

We argue that very few vertices in $x \in X$ and $y \in Y$ are heavily represented in round s of the sets of repeated game vertices $im(f_X)$ and $im(f_Y)$. Informally, we will use the uniform marginals property, which lets us conclude that there are many distinct base game edges (x, y) in round s of the edge set W, to conclude that there must be many distinct base game vertices in round s of endpoints of W. To formalize the notion of a vertex x being heavily represented in round s of $im(f_X)$, define the weight of $x \in X$ be $w_x = |\{\overline{x} \in im(f_X) : \overline{x}_s = x\}|$. Similarly, let the weight of $y \in Y$ be $w_y = |\{\overline{y} \in f_Y : \overline{y}_s = y\}|$. We argue that under the uniform marginals property of Theorem 4, there cannot be many vertices with weight more than $2z = 2\frac{|H|}{|G|}$.

▶ **Proposition 7.** *Take G and H to be the base game and repeated game from Theorem 4, and let E denote the edge set of the base game G. For any fixed edge $(x, y) \in E$ and round $j \in [k]$, we have that*

$$\left|\{(\overline{x}, \overline{y}) \in E_H : (\overline{x}_j, \overline{y}_j) = (x, y)\}\right| = z\,.$$

Proof. By applying the uniform marginals property of Theorem 4, we observe that

$$\begin{aligned}\left|\{(\overline{x}, \overline{y}) \in E_H : (\overline{x}_j, \overline{y}_j) = (x, y)\}\right| &= \Pr_{(\overline{x},\overline{y}) \in E_H}\left[(\overline{x}_j, \overline{y}_j) = (x, y)\right] \cdot |E_H| \\ &= \frac{|E_H|}{|E|} = z \qquad \blacktriangleleft\end{aligned}$$

Below, say that a repeated game vertex $v \in im(f_X) \cup im(f_Y)$ is BAD if v_s has weight more than $2z$.

▶ **Lemma 8.** *There are at most $2\varepsilon|E|$ repeated game edges in W incident to BAD vertices.*

For a fixed $x \in X$, define the set of vertices $PREIMG_x$ to be the set of edges in the repeated game that have x in the s^{th} coordinate, $PREIMG_x := \{\overline{x} \in im(f_X) : \overline{x}_s = x\}$. Notice that BAD vertices are exactly repeated game vertices that belong to $PREIMG_x$ for some $x \in X$ such that $w_x > 2z$. In what follows, we will let $d_X = d_Y = d$.

Proof (of Lemma 8). For each $x' \in X$ with weight $w_{x'} > 2z$, we will argue that there are many edges in $\widehat{W} \backslash W$ incident to $PREIMG_{x'}$. Then, by noticing that $\widehat{W}$ and W are ε-close, we will be able to upper bound the number of vertices in X with weight more than $2z$. Applying the uniform marginals property of H from Theorem 4, we will get an upper bound on the number of edges in W incident to BAD vertices.

Fix $x' \in X$ such that $w_{x'} > 2z$. We argue that there are at least $2z \cdot d_X$ edges in $\widehat{W}$ that are incident to $PREIMG_{x'}$. Recall that $\mathcal{H}_{\widehat{W}}$ is isomorphic to $\mathcal{G}$, the underlying graph of G. Specifically, this means that the degree of every member of $PREIMG_{x'}$ in $\mathcal{H}_{\widehat{W}}$ is exactly d_X, so in total there are $w_{x'} \cdot d_X \geq 2z \cdot d_X$ edges in $\widehat{W}$ incident to elements of $PREIMG_{x'}$.

Recall from Proposition 7 that the uniform marginals property of the scheme $\mathcal{S}$ tells us that, for any fixed edge $(x, y) \in E$, the number of edges $(\overline{x}, \overline{y}) \in E_H$ such that $(\overline{x}_s, \overline{y}_s) = (x, y)$ is exactly z. Since $W \subseteq E_H$, we conclude that for any fixed $(x, y) \in E$ we have that

$$|\{(\overline{x}, \overline{y}) \in W : (\overline{x}_s, \overline{y}_s) = (x, y)\}| \leq z\,.$$

By fixing x and summing over all y such that $(x, y) \in E$, of which there are exactly d_X, we can see that

$$|\{(\overline{x}, \overline{y}) \in W : \overline{x}_s = x\}| \leq z \cdot d_X$$

for any fixed $x \in X$. In other words, there can be at most $z \cdot d_X$ edges in W incident to vertices in $PREIMG_x$, for any $x \in X$.

Combining our lower bound of $2z \cdot d_X$ for the number of edges in $\widehat{W}$ incident to $PREIMG_{x'}$ and our upper bound of $z \cdot d_X$ for the number of edges in W incident to $PREIMG_{x'}$, we see there are at least $2z \cdot d_X - z \cdot d_X = z \cdot d_X$ edges in $\widehat{W} \backslash W$ that touch $PREIMG_{x'}$, and that this is true for all x' such that $w_{x'} > 2z$.

Noticing that there are not many edges in $\widehat{W} \backslash W$, we can upper bound the number of X vertices with weight more than $2z$. For each vertex $x \in X$, let the variable i_x denote the number of edges incident to the vertex set $PREIMG_x$ that are in $\widehat{W} \backslash W$. Since $|\widehat{W} \backslash W| \le \varepsilon|E|$, we get:

$$\begin{aligned}\varepsilon|E| &\ge \sum_{x \in X : w_x > 2z} i_x \\ &\ge |\{x \in X : w_x > 2z\}| \cdot z \cdot d_X\end{aligned}$$

So we get that the number of base game vertices $x \in X$ with weight more than $2z$ is at most $\frac{\varepsilon|E|}{zd_X}$. Reapplying the observation that there can be at most $z \cdot d_X$ edges incident to $PREIMG_x$ for any base game vertex x, we see that there can be at most $\varepsilon|E|$ edges in W incident to BAD vertices that live in $im(f_X)$.

Repeating the proof for vertices in Y shows there are at most $\frac{\varepsilon|E|}{zd_Y}$ vertices in Y with weight more than $2z$, and at most $\varepsilon|E|$ edges in W incident to BAD vertices that live in $im(f_Y)$. Union bounding over vertices in $im(f_X)$ and $im(f_Y)$ yields the result. ◀

Lemma 8 lets us remove all the bad vertices from $\mathcal{H}_W$, along with all the edges incident to them, and still have a graph with at least $(1 - 3\varepsilon)|E|$ edges. Call the resulting graph $\mathcal{H}'_W = ((X'_W, Y'_W), W')$. We remove the same vertices and the incident edges from $\mathcal{H}_{\widehat{W}}$ to get the graph $\mathcal{H}_{\widehat{W}'} = ((X'_W, Y'_W), \widehat{W}')$. Note that we still have $W' \subseteq \widehat{W}'$ and $|\widehat{W}' \backslash W'| \le \varepsilon|E|$, and since we did not remove many edges thanks to Lemma 8, we know that $|\widehat{W}'| \ge |W'| \ge (1 - 3\varepsilon)|E|$.

We would like to find a subset of vertices $S \subseteq X'_W$ such that every element of $\{x \in X : \exists \overline{x} \in S \text{ s.t. } \overline{x}_s = x\}$ have similar weights, and find an analogous subset $T \subseteq Y'_W$.

▶ **Lemma 9.** *There are subsets $S \subseteq X'_W$ and $T \subseteq Y'_W$ such that:*

1. $|\widehat{W}' \cap (S \times T)| \ge \frac{(1-6\varepsilon)|E|}{4 \log^2(z)}$
2. *$W' \cap (S \times T)$ is 2ε-close to $\widehat{W}' \cap (S \times T)$.*
3. *There are integers $w_x^*, w_y^* \in \mathbb{Z}^+$ such that for any $x \in X$ s.t. $\overline{x}_s = x$ for some $\overline{x} \in S$ and $y \in Y$ s.t. $\overline{y}_s = y$ for some $\overline{y} \in T$, we have that $w_x^* \le w_x \le 2w_x^*$ and $w_y^* \le w_y \le 2w_y^*$.*

Proof. For each pair of positive integers (i, j) such that $0 \le i, j \le \lceil \log(2z) \rceil - 1$, let $S_i = \{\overline{x} : \overline{x}_s = x \text{ for } x \in X \text{ s.t. } 2^i \le w_x \le 2^{i+1}\}$ and $T_j = \{\overline{y} : \overline{y}_s = y, y \in Y, 2^j \le w_y \le 2^{j+1}\}$. Note that the sets $\{\widehat{W}' \cap (S_i \times T_j) : 1 \le i, j \le \lceil \log(2z) \rceil - 1\}$ form a partition of the edges in $\widehat{W}'$, since we removed all BAD vertices and incident edges earlier. We will call a pair (i, j) bad if $W' \cap (S_i \times T_j)$ is more than 2ε-far from the edge set $\widehat{W}' \cap (S_i \times T_j)$, and good otherwise.

Since $\widehat{W}' \backslash W'$ has size at most $\varepsilon|E|$, we can upper bound the size of the set

$$\bigcup_{i,j:(i,j) \text{ is bad}} \widehat{W}' \cap (S_i \times T_j)$$

as follows:

$$\begin{aligned}
\varepsilon|E| &\geq |\widehat{W}'\backslash W'| \\
&= \sum_{0\leq i,j\leq \lceil \log(2z)\rceil -1} \left|\left(\widehat{W}'\cap(S_i\times T_j)\right)\setminus (W'\cap(S_i\times T_j))\right| \\
&\geq \sum_{(i,j):(i,j)\text{ is bad}} \left|\left(\widehat{W}'\cap(S_i\times T_j)\right)\setminus (W'\cap(S_i\times T_j))\right| \\
&\geq 2\varepsilon \sum_{(i,j):(i,j)\text{ is bad}} \left|\widehat{W}'\cap(S_i\times T_j)\right| \\
&= 2\varepsilon \left|\bigcup_{i,j:(i,j)\text{ is bad}} \widehat{W}'\cap(S_i\times T_j)\right|
\end{aligned}$$

Therefore, we can conclude that

$$\bigcup_{i,j:(i,j)\text{ is bad}} \widehat{W}'\cap(S_i\times T_j)$$

has at most $|E|/2$ edges, and therefore

$$\bigcup_{i,j:(i,j)\text{ is good}} \widehat{W}'\cap(S_i\times T_j)$$

has at least $(\frac{1}{2}-3\varepsilon)|E|$ edges. Since i and j range from 0 to $\lceil\log(2z)\rceil-1$, there are at most $(\log(2z)+1)^2 \leq 2\log^2(z)$ good pairs, so there is some choice of positive integers i^* and j^* such that $\widehat{W}'\cap(S_{i^*}\times T_{j^*})$ has at least $\frac{(1-6\varepsilon)|E|}{4\log^2(z)}$ edges and (i^*,j^*) is good, so $W'\cap(S_{i^*}\times T_{j^*})$ is 2ε-close to $\widehat{W}'\cap(S_{i^*}\times T_{j^*})$. Taking $S := S_{i^*}$, $T := T_{j^*}$, $w_x^* = 2^{i^*}$, and $w_y^* = 2^{j^*}$ completes the proof. ◀

Note that Property 2 of Lemma 9 allows us to lower bound the number of edges in $W'\cap(S\times T)$. Since $W'\cap(S\times T)$ is 2ε-close to $\widehat{W}'\cap(S\times T)$, we get that

$$|W'\cap(S\times T)| \geq (1-2\varepsilon)|\widehat{W}'\cap(S\times T)| \geq \frac{(1-2\varepsilon)(1-6\varepsilon)|E|}{4\log^2(z)}.$$

Furthermore, note that each vertex in X'_W has degree at most d_X in $\mathcal{H}_{\widehat{W}'}$, and furthermore each vertex in Y'_W has degree at most d_Y in $\mathcal{H}_{\widehat{W'}}$. This can be seen by noting that $\mathcal{H}_{\widehat{W}}$ is isomorphic to $\mathcal{G}$, and we removed some edges when we removed BAD vertices. Combining this with the fact that $|E| = |X|d_X = |Y|d_Y$ and applying the lower bound on $|\widehat{W}'\cap(S\times T)|$, we can lower bound the sizes of the vertex sets S and T from Lemma 9. Specifically, we get that $|S| \geq \frac{(1-6\varepsilon)|X|}{4\log^2(z)}$ and $|T| \geq \frac{(1-6\varepsilon)|Y|}{4\log^2(z)}$.

Now, in accordance with the proof outline, we would like to retrieve a large subset $M \subseteq X'_W$ such that, for all $\overline{x}, \overline{x}' \in M$ such that $\overline{x} \neq \overline{x}'$, we have that $\overline{x}_s \neq \overline{x}'_s$. Similarly, we want a large subset $N \subseteq Y'_W$ such that, for all $\overline{y}, \overline{y}' \in N$ such that $\overline{y} \neq \overline{y}'$, we have that $\overline{y}_s \neq \overline{y}'_s$.

▶ **Lemma 10.** *There are sets $M \subseteq X'_W$ and $N \subseteq Y'_W$ such that:*
1. *M contains at most one element of the set $\{\overline{x} \in X'_W : \overline{x}_s = x\}$ for any fixed $x \in X$. Also, N contains at most one element of the set $\{\overline{y} \in Y'_W : \overline{y}_s = y\}$ for any fixed $y \in Y$.*
2. *$W'\cap(M\times N)$ is 8ε-close to $\widehat{W}'\cap(M\times N)$*
3. *$|M| \geq \frac{(1-6\varepsilon)|X|}{8z\log^2(z)}$ and $|N| \geq \frac{(1-6\varepsilon)|Y|}{8z\log^2(z)}$*

Proof (of Lemma 10): Start with the sets $S \subseteq X'_W$ and $T \subseteq Y'_W$ as well as w_x^* and w_y^* from Lemma 9. We will show the existence of $M \subseteq S$ and $N \subseteq T$ with the desired properties. Set $w_x^{max} = \mathsf{min}(2\mathsf{w}_\mathsf{x}^*, 2\mathsf{z})$, and similarly set $w_y^{max} = \mathsf{min}(2\mathsf{w}_\mathsf{y}^*, 2\mathsf{z})$. We note that, for any $x \in X$ such that $\{\overline{x} \in S : \overline{x}_s = x\}$ is nonempty, we know that $w_x \leq w_x^{max}$, from Lemma 9 and our removal of BAD vertices, and we have a similar condition for vertices $y \in Y$.

For each vertex $x \in X$ such that $\{\overline{x} \in S : \overline{x}_s = x\}$ is nonempty, label each vertex in the set $\{\overline{x} \in S : \overline{x}_s = x\}$ as $x^1, \ldots, x^{w_x^{max}}$. Since $|\{\overline{x} \in S : \overline{x}_s = x\}| = w_x$ and $w_x^{max} \geq w_x$, each vertex gets a label. However, note that it is possible that $w_x^{max} > w_x$, in which case we wrap around with our labeling. Since Lemma 9 gives us that $w_x \geq w_x^* \geq w_x^{max}/2$, we know that any vertex receives at most two labels. Similarly, for each vertex $y \in Y$ such that $\{\overline{y} \in T : \overline{y}_s = y\}$ is nonempty, label each vertex in the set $\{\overline{y} \in T : \overline{y}_s = y\}$ as $y^1, \ldots, y^{w_y^{max}}$. Once again we observe that every vertex gets a label and any vertex receives at most two labels.

For $i \in \mathbb{Z}$ such that $1 \leq i \leq w_x^{max}$, let

$$M_i = \bigcup_{x \in X : \{\overline{x} \in S : \overline{x}_s = x\} \neq \emptyset} x^i .$$

Similarly, for $j \in \mathbb{Z}$ such that $1 \leq j \leq w_y^{max}$, let

$$N_j = \bigcup_{y \in Y : \{\overline{y} \in T : \overline{y}_s = y\} \neq \emptyset} y^j .$$

Since any vertex $\overline{x} \in S$ received at most two labels, note that it is present in M_i for at most two choices of i. Similarly, any vertex $\overline{y} \in T$ is present in N_j for at most two choices of j. Consider the sets of pairs of vertices given by

$$\left\{M_i \times N_j : 1 \leq i \leq w_x^{max}, 1 \leq j \leq w_y^{max}\right\}$$

The union of these sets contains $S \times T$. Hence, every edge in $W' \cap (S \times T)$ is in $W' \cap (M_i \times N_j)$ for some choice of i and j. Similarly, every edge in $\widehat{W'} \cap (S \times T)$ is in $\widehat{W'} \cap (M_i \times N_j)$ for some choice of i and j. Furthermore, as we noticed earlier, any vertex $\overline{x} \in S$ is in M_i for at most two choices of i and any vertex $\overline{y} \in T$ is in N_j for at most two choices of j. Therefore, any fixed pair of repeated game vertices $(\overline{x}, \overline{y})$ only appears in $M_i \times N_j$ for at most 4 choices of (i, j). Hence we know that

$$\sum_{i,j} |\widehat{W'} \cap (M_i \times N_j)| \geq \left|\bigcup_{i,j} \widehat{W'} \cap (M_i \times N_j)\right| \geq |\widehat{W'} \cap (S \times T)|$$

and that

$$\sum_{i,j} |(\widehat{W'} \backslash W') \cap (M_i \times N_j)| \leq 4|(\widehat{W'} \backslash W') \cap (S \times T)| \leq 8\varepsilon |\widehat{W'} \cap (S \times T)|$$

Therefore, the average fraction of edges in $\widehat{W'} \cap (M_i \times N_j)$ that are also in $(\widehat{W'} \backslash W') \cap (M_i \times N_j)$ is at most 8ε. Therefore, there must be a fixing of i^* and j^* such that the set of edges $W' \cap (M_{i^*} \times N_{j^*})$ is 8ε-close to $\widehat{W'} \cap (M_{i^*} \times N_{j^*})$, by pigeonhole. Furthermore, since $w_x^{max}, w_y^{max} \leq 2z$, we know that

$$|M_{i^*}| = \frac{|S|}{w_x^{max}} \geq \frac{|S|}{2z} \geq \frac{(1 - 6\varepsilon)|X|}{8z \log^2(z)}$$

and

$$|N_{j^*}| = \frac{|T|}{w_y^{max}} \geq \frac{|T|}{2z} \geq \frac{(1-6\varepsilon)|Y|}{8z\log^2(z)}$$

By letting $M := M_{i^*}$ and $N := N_{j^*}$, we conclude the proof. ◀

We notice that Lemma 10 also gives us an explicit lower bound on the number of edges in $W' \cap (M \times N)$. Since none of the vertices in M or N are BAD by construction, we know that

$$\widehat{W}' \cap (M \times N) = \widehat{W} \cap (M \times N) \tag{2}$$

since to get from $\widehat{W}$ to $\widehat{W}'$ we only removed edges incident to BAD vertices. Also recall that $\mathcal{H}_{\widehat{W}}$ is isomorphic to $\mathcal{G}$, and therefore has the same expansion property as G, given by the expansion property of Lemma 15. Since Property 3 of Lemma 10 lower bounds the size of M and N, we can apply the expansion property of $\mathcal{H}_{\widehat{W}}$ to get:

$$\left|\widehat{W} \cap (M \times N)\right| \geq (1-\varepsilon)\frac{d_X|M||N|}{|Y|} \tag{3}$$

By combining Item 2 of Lemma 10 and Equations 2 and 3, we see that

$$|W' \cap (M \times N)| \geq (1-8\varepsilon)\left|\widehat{W}' \cap (M \times N)\right| \geq (1-8\varepsilon)(1-\varepsilon)\frac{d_X|M||N|}{|Y|} \tag{4}$$

Now we can prove the main theorem.

Proof of Theorem 4. Take $M \subseteq X'_W$ and $N \subseteq Y'_W$ to be the sets given by Lemma 10. Let $M_s = \{\overline{x}_s : \overline{x} \in M\}$ and $N_s = \{\overline{y}_s : \overline{y} \in N\}$ be the sets that result from projecting the repeated game vertices in M and N onto round s. Due to Property 1 of Lemma 10, for every pair of vertices $\overline{x}^1, \overline{x}^2 \in M$, we know that $\overline{x}^1_s \neq \overline{x}^2_s$. Similarly, for every pair of vertices $\overline{y}^1, \overline{y}^2 \in N$, we know that $\overline{y}^1_s \neq \overline{y}^2_s$. Therefore, we see that

$$|M_s| = |M| \geq \frac{(1-6\varepsilon)|X|}{8z\log^2(z)}$$

and

$$|N_s| = |N| \geq \frac{(1-6\varepsilon)|Y|}{8z\log^2(z)}$$

where the lower bounds follow from Property (3) of Lemma 10. Furthermore, any assignment to vertices in M and N corresponds uniquely to an assignment to $M_s \subseteq X$ and $N_s \subseteq Y$, by simply restricting the assignment to vertices in M and N to round s.

Since $W' \cap (M \times N) \subseteq W_{\{s\}}$, we know that every edge in $W' \cap (M \times N)$ is satisfied in round s by the assignment ψ^H. By restricting ψ^H_X to M and ψ^H_Y to N, considering only round s of this assignment, and applying the fact that each edge in $W' \cap (M \times N)$ corresponds to a unique edge in round s, we retrieve an assignment that satisfies $|W' \cap (M \times N)|$ edges in the rectangular subgame $G_{M_s \times N_s}$. By applying the expansion property of $\mathcal{G}$, we can upper bound the number of edges in this rectangle:

$$|E \cap (M_s \times N_s)| \leq (1+\varepsilon)\frac{d_X|M||N|}{|Y|}$$

Hence, by applying Equation 4, the fraction of constraints in $G_{M_s \times N_s}$ satisfied by our assignment is at least

$$\frac{|W' \cap (M \times N)|}{|E \cap (M_s \times N_s)|} \geq \frac{(1-8\varepsilon)(1-\varepsilon)}{1+\varepsilon} > 1 - 11\varepsilon \geq \mathrm{val}(G) + \varepsilon$$

due to our assumption on $\mathrm{val}(G)$. This, along with the fact that

$$\begin{aligned}\delta &= \frac{1}{16\Phi_{\mathcal{S},\Sigma}(n)\log^2(\Phi_{\mathcal{S},\Sigma}(n))} \\ &\leq \frac{1}{16z\log^2(z)} \\ &\leq \frac{1-6\varepsilon}{8z\log^2(z)}\end{aligned}$$

means that we contradict the fact that G is (δ,ε)-fortified. ◀

Remark about handling Parallel Edges

We end this section by remarking on why parallel edges can be problematic and how we handle them. In the last step of the proof, we lift round s of the assignment ψ^H on the rectangle $M \times N \subseteq X^k \times Y^k$ to an assignment for the rectangular subgame $G_{M_s \times N_s}$. We argued that each edge in the edge set $W' \cap (M \times N)$ lifted to a distinct edge in $G_{M_s \times N_s}$, by virtue of the fact that each vertex in M and N is distinct in round s. This is valid when W' is a set of edges, rather than a multiset; however, if we considered W' to be a multiset and it had parallel edges, this may no longer be true. Two distinct, but parallel, edges in W', could lift to only one distinct edge in $G_{M_s \times N_s}$, in which case we lose an edge! In the case when the number of parallel edges is small (i.e. $\leq \varepsilon|E|$), we can prevent this inconvenience by effectively ignoring the parallel edges.

Concretely, we can make W a multiset that has no parallel edges by ignoring parallel edges in the domain of the embedding map (i.e. each pair of vertices that appears in W has multiplicity 1). Since the number of parallel edges is small, we will still have $|W| \geq (1-2\varepsilon)|E|$ and that W is 2ε-close to a multiset of edges $\widehat{W}$, where $\mathcal{H}_{\widehat{W}}$ is isomorphic to G, parallel edges and all. By naturally extending the notion of ε-closeness to multisets, and defining the intersection of a multiset and a set to preserve multiplicity (i.e. $\{1,1,1,2\} \cap \{1\} = \{1,1,1\}$), our arguments naturally extend to this case without any further change.

For completeness, we conclude with a note about the number of parallel edges in the random games we provide in Appendix D. As long as $200d^2 < \varepsilon|E|$, the random games we generate have sufficiently few parallel edges for our Main Theorem to apply. When $200d^2 > \varepsilon|E|$, since $|E| = d|X|$, we must have that $d = \Omega(|X|)$. For this regime of d, we can simply use a free game with random constraints. It can be seen by the analysis in Claim 19 of Appendix D that this game is sufficiently fortified and satisfies the conditions we need for the Main Theorem.

B Robust embeddings in existing proofs of parallel repetition

Here we show that Raz's proof of the parallel repetition theorem directly implies a robust embedding from G into G^k. Raz's proof was significantly simplified by Holenstein in [14]. Throughout this section, we will follow Rao's presentation [20] of Raz's proof with Holenstein's simplification. From now on, we will refer to this proof as the Raz-Holenstein proof of parallel repetition.

The engine behind the Raz-Holenstein proof of parallel repetition theorem is the following lemma.

▶ **Lemma 11** (Main lemma of [20]). *Let $C \subseteq [k]$. Let G be a game with $\mathrm{val}(G) = 1 - \varepsilon$, where one of the provers gives answers from a set of size 2^c, and there exists a strategy ψ for G^k under which*

$$\Pr[W_C] \geq 2^{-\frac{\varepsilon^2(k-|C|)}{34^2}+|C|c}.$$

Then there exists an $i \notin C$ such that $\Pr[W_i|W_C] \leq \mathrm{val}(G) + \varepsilon/2 = 1 - \varepsilon/2$.

Here, we use W_C to denote the event that the provers succeed in the rounds indexed by C; note that this event depends on the strategy used by the provers. We use $W_{\{i\}}$ to denote the event that the provers win round i.

From Lemma 11, the parallel repetition theorem follows in a straightforward manner. We want to show that the probability of winning every round in G^k, $\Pr[W_{[k]}]$, is $2^{-\gamma k}$ for some constant γ. We accomplish this by iteratively building a subset of rounds $C \subseteq [k]$ such that either $\Pr[W_C] < 2^{-\gamma k}$ (in which case we're done, because $\Pr[W_{[k]}] \leq \Pr[W_C]$), or otherwise, by upper bounding $\Pr[W_{\{i\}}|W_C]$ for some $i \notin C$, we conclude that $\Pr[W_{\{i\}\cup C}] < (1-\varepsilon/2)\Pr[W_C]$ and recurse with $C' = C \cup \{i\}$. After repeatedly applying this lemma at most βk times, we can conclude that $\Pr[W_{[k]}] \leq \max\{2^{-\gamma k}, (1-\varepsilon/2)^{\beta k}\}$, which proves the parallel repetition theorem.

Implicit in the proof of Lemma 11 is the following lemma, which demonstrates the existence of a robust embedding of G into G^k.

▶ **Lemma 12** (Implicit Lemma in [20]). *Let $C \subseteq [k]$ be such that*

$$\Pr[W_C] \geq 2^{-\frac{\varepsilon^2(k-|C|)}{34^2}+|C|\cdot c}.$$

Then there exist randomized maps $g_X : R \times X \to X^k$ and $g_Y : R \times Y \to Y^k$ for some finite set R such that

1. *For all $r \in R$, there exists a round $i \in [k] \backslash C$ such that for all $(x, y) \in X \times Y$, we have $g_X(r, x)_i = x$ and $g_Y(r, y)_i = y$.*
2. *The distribution of $(g_X(r, x), g_Y(r, y))$ over a uniformly chosen $r \in R$ and $(x, y) \in E$ is $\varepsilon/2$-close in statistical distance to the distribution of $(\overline{x}, \overline{y})$ in G^k when conditioned on the event W_C.*

First, we claim that Implicit Lemma very directly implies the existence of a robust embedding from G into a coordinate of G^k. Indeed, assume that Lemma 12 is true. Let $\gamma : [k] \to \mathbb{R}$ be defined as $\gamma(t) = 2^{-\frac{\varepsilon^2(k-t)}{34^2}+t\cdot c}$. For each C, if $\Pr[W_C] < \gamma(|C|)$, then we let Emb be an arbitrary embedding map. If $\Pr[W_C] \geq \gamma(|C|)$, then there exist randomized maps g_X and g_Y satisfying Property 2 of Lemma 12. Furthermore, by averaging, there must exist an $r^* \in R$ such that $\Pr_{(x,y)\in E}[(g_X(r^*, x), g_Y(r^*, y)) \in W_C] \geq 1 - \varepsilon/2$. Let $\mathsf{Emb}(x, y) = (g_X(r^*, x), g_Y(r^*, y))$. This shows that there is a $(\gamma(t), \varepsilon/2)$-robust embedding of G into a coordinate of G^k.

Furthermore, the Implicit Lemma also implies Lemma 11:

Proof that Lemma 12 Implies Lemma 11. We assume the Implicit Lemma. Let $C \subseteq [k]$ be as described in the statement of the lemma, and let $g_X : R \times X \to X^k$ and $g_Y : R \times Y \to Y^k$ be the randomized embedding maps.

We now describe a strategy for the provers to play the base game G. The provers are given x and y where (x, y) is a uniform edge from E. The two provers, using shared randomness, sample a uniformly random $r \in R$. The first prover computes $\overline{x} = g_X(r, x)$ and then $\overline{a} = \psi_X(\overline{x})$. The first prover answers with $\overline{a}_i$. The second prover computes $\overline{y} = g_Y(r, y)$ and then $\overline{b} = \psi_Y(\overline{y})$. The second prover answers with $\overline{b}_i$.

Since this is a strategy for G, the probability that the provers win is at most $\mathrm{val}(G)$. On the other hand, since the distribution of $(\overline{x}, \overline{y})$ generated by the provers is $\varepsilon/2$-close to the distribution of $(\overline{x}_i, \overline{y}_i)$ in the subgame W_C, we have that the probability the provers win using this strategy is at least $\Pr[W_i|W_C] - \varepsilon/2$.

Thus we have $\Pr[W_i|W_C] \geq \mathrm{val}(G) + \varepsilon/2$. ◀

Finally, for completeness, we give a high-level sketch of how the Implicit Lemma is proved. This argument follows the Raz-Holenstein proof of the parallel repetition theorem. Let $C \subseteq [k]$ be a set of coordinates such that $\Pr[W_C] \geq \gamma(|C|)$. Let $\overline{X}, \overline{Y}, \overline{A}, \overline{B}$ denote the random variables corresponding to the questions and answers of the provers when playing G^k. The randomized maps g_X and g_Y will correspond to a protocol where the first prover (who receives a question $x \in X$) and the second prover (who receives $y \in Y$) utilize shared randomness R in order to agree on a coordinate $i \in [k] \backslash C$, and produce questions $\overline{x} \in X^k$ and $\overline{y} \in Y^k$, respectively, so that $\overline{x}_i = x$, $\overline{y}_i = y$, and furthermore, their outputs $(\overline{x}, \overline{y})$ are (approximately) distributed the same way as $(\overline{X}, \overline{Y})$ are, conditioned on the event W_C.

The key to this protocol, and the cornerstone of the Raz-Holenstein parallel repetition theorem is the *dependency-breaking random variable* Q, which resides in the same probability space as $\overline{X}, \overline{Y}, \overline{A}, \overline{B}$. This random variable has the property that, conditioned on Q and (say) the first prover's question x, the repeated questions $\overline{X}$ and $\overline{Y}$ are *independent*. Furthermore, the variable Q has the remarkable property that the following distributions are close in statistical distance[4]:

$$p(Q|\overline{X}_i = x, W_C) \approx p(Q|\overline{X}_i = x, \overline{Y}_i = y, W_C) \approx p(Q|\overline{Y}_i = y, W_C)$$

where by $p(Q|\overline{X}_i = x, W_C)$, for example, we mean the distribution of Q conditioned on $\overline{X}_i = x$ and the event W_C. Using a beautiful technique called *correlated sampling*, the two provers can use shared randomness to (approximately) jointly sample Q from the distribution $p(Q|\overline{X}_i = x, \overline{Y}_i = y, W_C)$, even though they only know one of x or y, but not both.

Since i was picked randomly, with high probability the distribution of $(\overline{X}_i, \overline{Y}_i)$ conditioned on W_C will also be close to the distribution of questions in the original game G. This implies that the final distribution of the output of the maps g_X and g_Y will be close to the distribution of $(\overline{X}, \overline{Y})$ conditioned on W_C, which is what we desired.

In addition to the Raz-Holenstein proof, nearly all subsequent proofs of parallel repetition fall into the embedding framework, including the works of Rao [20], Moshkovitz [17], and Braverman-Garg [7]. We also believe that the analytical proof of parallel repetition given by Dinur and Steurer in [11] falls under this framework.

C A Contrived Example for Derandomized Parallel Repetition

In this section we show that we cannot hope to obtain a strong no-go theorem that rules out any derandomized parallel repetition in the high degree regime, the same spirit as the result of Feige and Kilian. This is because there *is* a parallel repetition scheme that, when applied to some games, actually reduces the value in a very randomness-efficient manner. We use Dinur's graph powering gap amplification scheme, which is a highly randomness-efficient

[4] Technically speaking, they are close on average over i, x, and y.

parallel repetition scheme. For any $\varepsilon > 0$, we construct a game G with value $\geq 1/8$, such that the application of graph powering to G yields a game H with value at most ε, and the randomness complexity of H is $\log|G| + f(1/\varepsilon)$ for some function f. If $|G|$ is a growing parameter, then for constant ε, this is much less than $O(\log \frac{1}{\varepsilon}) \cdot \log|G|$, the randomness complexity that would be needed if we used standard parallel repetition to reduce the value from $1/8$ to ε.

Unfortunately, this doesn't show that graph powering is a useful derandomized parallel repetition scheme[5]. The game G is constructed by first taking a game G_{low} with value ε, and "hiding" it in a high value game G with value at least $1/8$. The game H produced by graph powering "uncovers" G_{low}, and thus $\text{val}(H) \leq \text{val}(G_{low}) \leq \varepsilon$. However, intuitively the error reduction was not obtained by graph powering per se, but rather came from a "planted" game that had low value to begin with. This shows that the degree-dependent lower bound of Feige and Kilian is in a sense tight, and thus to obtain stronger no-go results for derandomized parallel repetition, we turn to investigating proof strategies, which is the focus of our paper.

C.1 The Derandomized Parallel Repetition Scheme: Graph Powering

Specifically, the derandomized parallel repetition scheme we use is graph powering, well-known from the gap amplification scheme of Dinur. This transforms a graph $G = (V, E)$ to a graph $G^{*t} = (V', E')$. In this graph, we have that $V' = V$, and each vertex $v \in V'$ intuitively corresponds to the "cloud" of vertices reachable from $v \in V$ in t steps. Furthermore, each edge in E' corresponds to a $(2t+1)$-step random walk in E. The prover is supposed to give each vertex $v' \in V'$ a super-label that contains labels for each of the vertices in its "cloud," and each edge $e' = (u', v') \in E'$ checks that: 1) the labels to u' and v' are valid and 2) there is consistency in the labels of all the vertices shared between the cloud of u' and the cloud of v'.

The graph powering method described above is a form of derandomized parallel repetition. If we let d denote the maximum degree of the graph G, selecting a random edge in G^{*t} takes $\log|V| + (2t+1)\log d$ bits of randomness, as edges in G^{*t} are simply $(2t+1)$-length random walks. Note that with t and d being constant, this is an extremely randomness efficient way to ask many questions. The main problem with using this as a derandomized parallel repetition scheme is that it is unclear how to prove that the value of G^{*t} is decaying with increasing t. However, in this section we will create games G for which the value of G^{*t} is significantly lower than the value of G, and hence be able to use graph powering as derandomized parallel repetition. In fact, we will only need to focus on the case where $t = 2$: that is, in this section, we will construct games G where the value of G^{*2} is much lower than the value of G.

We also observe that the alphabet size of G^{*t} is $|\Sigma|^{d^t}$ (since we are asking for labels to all the vertices reachable in t steps from a vertex v). In our construction, $|\Sigma|$, d, and t are all constant (relative to the size of the game), and thus the alphabet size is constant.

C.2 A Sketch of the Construction

The rough outline of the construction is as follows:

[5] In fact, Bogdanov constructs games for which graph powering fails to achieve any error reduction at all [6].

1. Start with a two prover game $G' = (X', Y', E', \Sigma, \mathcal{C})$, where $\mathcal{C}$ denotes the constraints on the game G', that has low constant value ε, has a constant sized alphabet, and has constant degree.
2. Use *composition* to transform G' into a game G over the alphabet $\{0,1\}^3$. An exposition on composition can be found in Section 5 of [9]. Roughly speaking, by composition we mean that we replace each constraint in the game G' with a gadget that encodes the constraint, but is itself a game over alphabet $\{0,1\}^3$. Such gadgets are called *assignment testers*, and have a size that depends only on the alphabet size of G'. The game G that we get after composition necessarily has high value, as a random strategy satisfies at least 1/8th of the constraints. More details can be found in Section C.4 below.
3. Use graph powering on G to get the game G^{*2}, which will have value at most that of G'.

This construction works by using composition to hide the low value game G' inside the high-value game G. However, the hiding was performed in a local fashion that can easily be uncovered by graph powering. Namely, the game G^{*2} will contain constraints of G', and hence have low value. Furthermore, due to the constant degree and alphabet size of G', the game G^{*2} will have very low randomness complexity – no more than the randomness complexity of G plus an additive constant. We now go into each step in further detail.

C.3 Step 1: A Game with Low Value

We start with a two player game $G' = (X', Y', E', \Sigma, \mathcal{C})$ with $\text{val}(G') < \varepsilon$, and the alphabet size and degree are functions of $1/\varepsilon$. Since we think of ε as a constant, the alphabet size and degree are also constant.

C.4 Step 2: Composition

Recall that our goal in this section is to transform the game G' into a game G over the alphabet $\{0,1\}^3$. For this we will use composition with *assignment testers* as described in Definition 5.1 in [9]. We define assignment testers below:

▶ **Definition 13** (Assignment Tester, Definition 2.2 from [9])**.** An Assignment Tester with alphabet Σ_0 and rejection probability $\varepsilon > 0$ is an algorithm $\mathcal{P}$ whose input is a circuit Φ over Boolean variables X, and whose output is a constraint graph $G = ((V, E), \Sigma_0, \mathcal{C})$ such that $V \supset X$ and the following hold. Let $V' = V \backslash X$, and let $a : X \to \{0,1\}$ be an assignment.
- (Completeness) If $a \in \text{SAT}(\Phi)$, there exists $b : V' \to \Sigma_0$ such that $\text{UNSAT}_{a \cup b}(G) = 0$.
- (Soundness) If $a \notin \text{SAT}(\Phi)$, then for all $b : V' \to \Sigma_0$, we have $\text{UNSAT}_{a\cup b}(G) \geq \varepsilon \cdot \text{rdist}(a, \text{SAT}(\Phi))$.

where $\text{rdist}(a, S) = \min_{s\in S} \frac{|a \oplus s|}{|V|}$ denotes the minimum relative Hamming distance between a and elements of the set S, $\text{SAT}(\Phi)$ is the set of satisfying inputs to Φ, and $\text{UNSAT}_{a\cup b}(G)$ is the fraction of constraints of G that are unsatisfied by the assignment induced by a and b.

Additionally, Theorem 5.1 of [9] gives us that there are explicit assignment testers over $\{0,1\}^3$ for a certain $\varepsilon > 0$.

Using assignment testers, we can describe the composition of a game G and an assignment tester $\mathcal{P}$. For this, we will use an error correcting code $e : \Sigma \to \{0,1\}^\ell$, where $\log_2 |\Sigma| \leq \ell \leq c \cdot \log_2 |\Sigma|$ for some constant c.

▶ **Definition 14** (Composition, Definition 5.1 from [9])**.** Let $G = ((V, E), \Sigma, \mathcal{C})$ be a constraint graph and let $\mathcal{P}$ be an assignment tester. Let $e : \Sigma \to \{0,1\}^\ell$ be an encoding as described above with relative distance $\rho > 0$. The constraint graph $G \circ \mathcal{P} = ((V', E'), \Sigma_0, \mathcal{C}')$ is defined in two steps:

- (Robustization): First, we convert each constraint $c(e) \in \mathcal{C}$ to a circuit $\tilde{c}(e)$ as follows. For each variable $v \in V$, let $[v]$ be a fresh set of ℓ Boolean variables. For each edge $e = (v, w) \in E$, $\tilde{c}(e)$ will be a circuit on 2ℓ Boolean variables $[v] \cup [w]$ that outputs 1 iff the assignment for $[v] \cup [w]$ is a legal assignment for v and w that would have satisfied the constraint c on (v, w).
- (Composition): Run the assignment tester $\mathcal{P}$ on each $\tilde{c}(e)$. Let $G_e = ((V_e, E_e), \Sigma_0, \mathcal{C}(e))$ denote the resulting constraint graph, and recall that $[v] \cup [w] \subset V_e$. Assume, wlog, that E_e has the same cardinality for each e. Define the new constraint graph $G \circ \mathcal{P} = ((V', E'), \Sigma_0, \mathcal{C}')$ by

$$V' = \bigcup_{e \in E} V_e \qquad E' = \bigcup_{e \in E} E_e \qquad \mathcal{C}' = \bigcup_{e \in E} \mathcal{C}_e$$

As noted in [9], the output graph G_e of an assignment tester $\mathcal{P}$ when it is used in composition above has size that depends only on the alphabet size of the game G', which is a constant. Hence, the size of G_e is also a constant. Furthermore, it can be seen from Definitions 13 and 14 that $G_{(u,v)}$ can have all its constraints satisfied if and only if the assignments given to $[u]$ and $[v]$ are legal assignments for u and v that satisfy the constraint $c((u, v))$.

We will consider the modified assignment tester $\mathcal{P}'$, which acts as follows. It runs $\mathcal{P}$ on the input, and looks at the resulting constraint graph H. It then adds all missing edges to H to create a complete graph $\overline{H}$, and puts trivially satisfied constraints on all of them. It can be seen that if H had constant size, then so does $\overline{H}$. Note that the constraints of H are all satisfiable if and only if the constraints of $\overline{H}$ are all satisfiable. Hence, the output graphs of the assignment tester $\mathcal{P}'$ also satisfy the property that all of its constraints are satisfiable if and only if the input variables encoded a satisfying a legal and satisfying assignment to the input constraint.

We will define the constraint graph G as $G' \circ \mathcal{P}'$. The high connectivity of each gadget $\overline{H}$ will be very useful to us in Step 3.

This process gives us a constraint graph G with $\text{val}(G) \geq 1/8$, since a random strategy can achieve $\text{val}(G) \geq 1/8$ in games over an alphabet of size 8.

C.5 Step 3: Randomness-Efficient Parallel Repetition via Graph Powering

Fix a vertex v in the game G. This vertex lies in $G_{(u',w')}$ for some $(u', w') \in E'$, where $G_{(u',w')}$ denotes the output of the assignment tester $\mathcal{P}'$ on $[u']$ and $[w']$. Now consider the graph G^{*2}. The label to v in G^{*2} claims labels to all vertices in $G_{(u',w')}$ due to the fact that $G_{(u',w')}$ is a complete graph. This label is valid if and only if all the constraints in $G_{(u',w')}$ are satisfied, which occurs if and only if the labels to $[u']$ and $[w']$ encode valid and satisfying labels for the edge $(u', w') \in E'$. Therefore, even picking a uniform vertex in G^{*2} and testing the validity of its label already performs a uniform test in G', and hence $\text{val}(G^{*2}) \leq \text{val}(G') < \varepsilon$.

As discussed in Section C.1, the amount of randomness used to sample a random constraint in G^{*2} consists of the randomness to query a single vertex of G^{*2}, which consists the randomness required to select a single vertex of G, and the randomness required to take a two step random walk in G. The degree of G is a function of two things: the size of the output graphs of the assignment testers and the degree of G'. Both of these are constant in our setting, and so taking a two step walk on G takes constant amount of randomness. Hence, using a derandomized parallel repetition scheme, we can transform a game G with $\text{val}(G) \geq 1/8$ to a game G^{*2} with $\text{val}(G^{*2}) < \varepsilon$ for an arbitrarily small constant ε, where the size of G^{*2} is $|G^{*2}| = c(\Sigma, d)|G|$, and Σ and d denote the alphabet size and degree of the game G'. Since Σ and d are functions of $1/\varepsilon$, for constant ε these are also constant.

We note that to get soundness ε will normal parallel repetition, we would have had to repeat the game at least $k = \log_8 \frac{1}{\varepsilon}$ times, and so the size of this game G^k would be $|G|^k = \Omega(|G|^{\log \frac{1}{\varepsilon}})$. We can see that G^{*2} is considerably smaller than this, and is in fact almost-linear in $|G|$.

D Random games are fortified

In this section we prove that randomly sampled d-regular bipartite graphs are fortified with high probability, and can therefore be used as input games to the Main Theorem. Formally, we prove the following:

▶ **Lemma 15.** *Let $0 < \eta, \delta < 1$. Let $0 < \beta < 1/2$. Let t be an integer and let Σ be a finite alphabet. Let $d > \frac{4(1+\ln|\Sigma|)}{\eta^2\delta^2}$. Let $\mathcal{G} = ([t] \times [t], E)$ be a bipartite graph that is the union of d random perfect matchings $M^1, \dots, M^d$, and let $G = (X, Y, E, \pi, \Sigma)$ be a game where $X = Y = [t]$ and for each edge $e \in E$, π_e is a randomly chosen subset of $\Sigma \times \Sigma$ of density β. Then the following properties hold with probability at least .99:*

1. *$\mathcal{G}$ is d-regular, and has at most $200d^2$ parallel edges.*
2. *For all $S, T \subseteq [t]$ with $|S|, |T| \geq \delta t$, we have*
$$\left|\frac{|E \cap (S \times T)|}{|S||T|} - \frac{d}{t}\right| \leq \eta\frac{d}{t}.$$
3. *$\mathrm{val}(G) \leq \beta + \eta$.*
4. *G is $(\delta, 2\eta)$-fortified.*

Note that, if we set $\varepsilon = 2\eta$ and assume that $200d^2 < \varepsilon|E|$, the games provided by Lemma 15 satisfy the conditions we require in Theorem 4. Before proving the lemma, we prove a general lemma about the sampling properties of d random perfect bipartite matchings.

▶ **Lemma 16** (Random matchings sample well)**.** *Let $M^1, \dots, M^d$ be d perfect matchings on $[t] \times [t]$ sampled uniformly at random. Let $Z \subseteq [t] \times [t]$ be an arbitrary set, and let $\mu = |Z|/t^2$. Then with probability at least $1 - \exp(-\Omega(\rho^2\mu^2 dt))$, $\left||\bigcup_j M^j \cap Z| - \mu dt\right| \leq \rho \cdot \mu dt$.*

Proof. We treat the selection of a random matching M^j as a result of a random process where first, the edges of the complete bipartite graph $K_{t,t}$ are ordered randomly, and then the edges in $M^j \subset K_{t,t}$ are revealed one by one according to this random order. Let E_i^j denote the ith revealed edge in M^j. Let Y_i^j be the indicator variable for whether $E_i^j \in Z$. Let $Y = \sum_j \sum_i Y_i^j$. Imagine a random process that first reveals all the edges of M^1 one at a time, then all the edges of M^2 one at a time, and so forth. Define a sequence of $td + 1$ random variables $X_0, X_1^1, \dots, X_t^1, X_1^2, \dots, X_t^2, \dots, X_1^d, \dots, X_t^d$, where $X_0 = \mathbb{E}[Y]$ and

$$X_i^j = \mathbb{E}[Y \mid E_{\leq(j,i)}]$$

where $E_{\leq(j,i)}$ denotes the sequence $E_1^1, \dots, E_t^{j-1}, E_1^j, \dots, E_i^j$, i.e., all the edges in matchings $M^1, \dots, M^{j-1}$, and the first i edges in matching M^j. By construction, the random variable sequence $\{X_i^j\}$ forms a Doob martingale with respect to the sequence $\{E_i^j\}$. We wish to apply Azuma's inequality to this to show that Y is tightly concentrated about its mean, which is

$$X_0 = \mathbb{E}[Y] = \sum_j \sum_i \mathbb{E}[Y_i] = \mu dt,$$

by linearity of expectation and the fact that the marginal distribution on each edge of M^j is a uniformly random edge in $K_{t,t}$. In order to apply Azuma's inequality, we need to establish that $\max\{|X_i^j - X_{i-1}^j|, |X_1^j - X_n^{j-1}|\} < c$ for some constant c. We argue that $c = 4$.

We introduce some notation that will be useful for us. Let U_i^j denote the complete bipartite graph on all the vertices that haven't been "paired" up by the edges $E_1^j, \ldots, E_i^j$. In other words, it is the subgraph of $K_{t,t}$ where the edges $E_1^j, \ldots, E_i^j$, and all adjacent edges to them are removed. Let $\mathcal{M}_i^j$ denote the set of all perfect matchings on U_i^j. Note that, for all i, the matching M^j is contained in $\mathcal{M}_i^j$. We will let U_0^j denote $K_{t,t}$ and $\mathcal{M}_0^j$ to simply be the set of all perfect matchings on $K_{t,t}$. Finally, for all matchings (not necessarily perfect) M of $K_{t,t}$, let $\alpha(M)$ denote $|M \cap Z|$.

Consider the difference $|X_1^j - X_n^{j-1}|$. Suppose that the edges in the sequence $E_{<(j,1)}$ – i.e., all the edges in matchings $M^1, \ldots, M^{j-1}$ – have been revealed. Then we have

$$\begin{aligned}
X_1^j - X_t^{j-1} &= \mathbb{E}[Y \mid E_{\le(j,1)}] - \mathbb{E}[Y \mid E_{<(j,1)}] \\
&= \sum_{j' \ge j} \mathbb{E}\left[\sum_i Y_i^{j'} \middle| E_{\le(j,1)}\right] - \mathbb{E}\left[\sum_i Y_i^{j'} \middle| E_{<(j,1)}\right] \\
&= \mathbb{E}\left[\sum_i Y_i^{j} \middle| E_{\le(j,1)}\right] - \mathbb{E}\left[\sum_i Y_i^{j} \middle| E_{<(j,1)}\right] \\
&= \mathbb{E}\left[\sum_i Y_i^{j} \middle| E_1^j\right] - \mathbb{E}\left[\sum_i Y_i^{j}\right]
\end{aligned}$$

In the second line we used the linearity of expectation and the fact that, conditioned on $E_{<(j,1)}$, the random variables $Y_{i'}^{j'}$ are all fixed (i.e. revealing more edges from other matchings do not change their values) for all i' and all $j' < j$. In the third line, we use that revealing an edge in matching M^j does not affect the random variables $Y_i^{j'}$ for $j' > j$. We use the same reasoning in the fourth line; Y_i^j is independent of the edges of $M^1, \ldots, M^{j-1}$.

Observe that, conditioned on E_1^j, we have that M^j is a uniformly distributed matching in $\mathcal{M}_1^j$ adjoined with E_1^j (since $\mathcal{M}_1^j$ technically contains submatchings). Without conditioning on E_1^j, M^j is a uniformly distributed matching in $\mathcal{M}_0^j$. Thus we have the identities

$$\mathbb{E}\left[\sum_i Y_i^j \middle| E_1^j\right] = Y_1^j + \left(\frac{1}{|\mathcal{M}_1^j|} \sum_{N \in \mathcal{M}_1^j} \alpha(N)\right)$$

and

$$\mathbb{E}\left[\sum_i Y_i^j\right] = \frac{1}{|\mathcal{M}_0^j|} \sum_{M \in \mathcal{M}_0^j} \alpha(M).$$

Define the mapping $B : \mathcal{M}_0^j \to \mathcal{M}_1^j$ on matchings where, for all matchings $M \in \mathcal{M}_0^j$:

- If M contains E_1^j, then $B(M)$ is the submatching M restricted to U_1^j.
- Else if M contains (a, d) and (c, b) where $E_1^j = (a, b)$, then $B(M)$ is the submatching M restricted to U_1^j adjoined with (c, d) (which was not in originally in M).

Fix an $M \in \mathcal{M}_0^j$. Suppose that $E_1^j \in M$. Then $|\alpha(M) - \alpha(B(M))| \le 1$. Otherwise, $|\alpha(M) - \alpha(B(M))| \le 2$, because it could be that both (a, d) and (b, c) are in Z, and (c, d) is not.

Furthermore, observe that the map B is onto, and for all $N \in \mathcal{M}_1^j$, the sizes of the preimages $B^{-1}(N) \subset \mathcal{M}_0^j$ are all the same. Then we have

$$\mathbb{E}\left[\sum_i Y_i^j\right] = \frac{1}{|\mathcal{M}_1^j|} \sum_{N \in \mathcal{M}_1^j} \frac{|\mathcal{M}_1^j|}{|\mathcal{M}_0^j|} \sum_{M \in B^{-1}(N)} \alpha(M)$$

so

$$\begin{aligned}
\left|X_1^j - X_n^{j-1}\right| &= \left|\mathbb{E}\left[\sum_i Y_i^j \middle| E_1^j\right] - \mathbb{E}\left[\sum_i Y_i^j\right]\right| \\
&\leq |Y_1^j| + \left|\frac{1}{|\mathcal{M}_1^j|} \sum_{N \in \mathcal{M}_1^j} \left(\alpha(N) - \frac{|\mathcal{M}_1^j|}{|\mathcal{M}_0^j|} \sum_{M \in B^{-1}(N)} \alpha(M)\right)\right| \\
&\leq 1 + \frac{1}{|\mathcal{M}_1^j|} \sum_{N \in \mathcal{M}_1^j} \frac{|\mathcal{M}_1^j|}{|\mathcal{M}_0^j|} \sum_{M \in B^{-1}(N)} |\alpha(B(M)) - \alpha(M)| \\
&\leq 3.
\end{aligned}$$

The first inequality follows from triangle inequality, the second inequality follows from the fact that the number of $M \in B^{-1}(N)$ is equal to $|\mathcal{M}_0^j|/|\mathcal{M}_1^j|$, and the third inequality follows from our bound on the difference $|\alpha(B(M)) - \alpha(M)|$.

Since this holds for every fixing of $E_{<(j,1)}$, this implies that $|X_1^j - X_t^{j-1}| < 4$ with certainty. The same argument as above also implies that for all i, $|X_{i+1}^j - X_i^j| < 4$ with certainty. Hence, we can apply Azuma's inequality:

$$\Pr(|X_n^d - X_0| \geq \rho \cdot \mu dt) \leq 2 \exp\left(-\frac{\rho^2 \mu^2 dt}{2c^2}\right).$$

We conclude the theorem by observing that X_n^d is the number of edges in the union of the matchings $M^1, \ldots, M^d$ that fall within Z. ◀

▶ **Lemma 17.** *Let $\mathcal{G} = ([t] \times [t], E)$ be a bipartite graph that is the union of d random perfect matchings on $[t] \times [t]$. Then the probability that there are more than $200d^2$ parallel edges in E is less than $1/200$.*

Proof. Let $M^1, \ldots, M^d$ denote the matchings. For $1 \leq j, j' \leq d$, and $1 \leq i \leq n$, let $X_{j,j',i}$ denote the indicator variable that the ith left vertex gets matched to the same right vertex under matchings M^j and $M^{j'}$. Note that $\mathbb{E}[X_{j,j',i}] = 1/t$. Note that the number of parallel edges is at most $\sum_{j,j'} \sum_i X_{j,j,i'}$, and thus the expected number of parallel edges is at most d^2. By Markov's inequality, the number of parallel edges is at most $200d^2$ with probability at least $1 - 1/200$. ◀

▶ **Corollary 18.** *Let $0 < \delta, \rho < 1$, and let $d > 1/(\rho^2\delta^2) + 2$. Let $\mathcal{G} = ([t] \times [t], E)$ be a bipartite graph that is the union of d random perfect matchings on $[t] \times [t]$. Then with probability at least $1 - \exp(-\Omega(\rho^2\delta^2 dt))$, for every $S, T \subseteq [t]$ where $|S|, |T| \geq \delta t$, we have that*

$$\left|\frac{|E \cap (S \times T)|}{|S||T|} - \frac{d}{t}\right| \leq \rho \frac{d}{t}.$$

Proof. This follows from Lemma 16 and union bounding over all $S, T \subseteq [t]$ such that $|S|, |T| \geq \delta t$ (of which there are at most 2^{2t}). ◀

We now prove Lemma 15, which we restate here for completeness.

of Lemma 15. By Lemma 17 and Corollary 18, we have that with probability at least $199/200 - \exp(-\Omega(\rho^2\delta^2 dt)) \geq 198/200$, the graph $\mathcal{G}$ is such that properties (1) and (2) of the lemma statement are satisfied. Call this event H.

We now argue that properties (3) and (4) are satisfied with high probability, conditioned on H. Define $m := td$.

▶ **Claim 19.** *Let $S \subseteq X$ and $T \subseteq Y$ be such that $|S|, |T| \geq \delta t$. The probability that there exist assignments $\psi_X : X \to \Sigma$ and $\psi_Y : Y \to \Sigma$ such that more than 2η fraction of the constraints π_e such that $e \in E \cap (S \times T)$ are satisfied by (ψ_X, ψ_Y), conditioned on H, is at most* $\exp(-(\eta^2\delta^2 d - 2(\ln|\Sigma|))t)$.

Proof. Fix $\psi_X : X \to \Sigma$ and $\psi_Y : Y \to \Sigma$. Let $E_{S\times T}$ denote $E \cap (S \times T)$. We have that $|E_{S\times T}| \geq \delta td/2$. Given a fixed assignment, the probability a randomly chosen constraint π_e for an edge $e \in E_{S\times T}$ is satisfied by the assignment is β. Thus the expected fraction of satisfied edges is $\beta|E_{S,T}|$. By Chernoff, the probability that more than $(\beta + \eta)|E_{S\times T}|$, or less than $(\beta - \eta)|E_{S\times T}|$ edges are satisfied is at most $\exp(-2\eta^2|E_{S\times T}|) \leq \exp(-\eta^2\delta^2 m)$ by our condition on the size of $E_{S\times T}$.

Union bounding over all $|\Sigma|^{2t} = \exp(2(\ln|\Sigma|)t)$ possible assignments (ψ_X, ψ_Y), we have that the probability that there exists an assignment such that more than $2\eta|E_{S\times T}|$ edges are satisfied is at most $\exp(-(\eta^2\delta^2 m - 2(\ln|\Sigma|)t))$. ◀

Let $J_{S,T}$ denote the event that for all assignments (ψ_X, ψ_Y), no more than $\beta + \eta$ fraction of edges in $E \cap (S \times T)$ are satisfied by (ψ_X, ψ_Y). Let J denote the event that $J_{S,T}$ holds for all S, T of size at least δn. By union bound, the probability that J does not hold is at most

$$2^{2t} \cdot \exp(-(\eta^2\delta^2 m - 2(\ln|\Sigma|)t)) = \exp(-(\eta^2\delta^2 m - 2(1 + \ln|\Sigma|)t)).$$

Since

$$d > \max\left\{\frac{2}{\delta}\ln\frac{1}{\delta}, \frac{4(1+\ln|\Sigma|)}{\eta^2\delta^2}\right\}$$

then the probability that J and H both do not hold is at most

$$\Pr(\neg H) + \Pr(\neg J|H) \leq .99.$$

But if J and H both hold, this implies that for all S, T of size at least δt, the fraction of satisfiable edges is at between $\beta - \eta$ and $\beta + \eta$. Thus this implies that G is $(\delta, 2\eta)$-fortified. ◀

Fast Synchronization of Random Automata

Cyril Nicaud[*]

Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, Marne-la-Vallée, France
cyril.nicaud@u-pem.fr

Abstract

A synchronizing word for an automaton is a word that brings that automaton into one and the same state, regardless of the starting position. Černý conjectured in 1964 that if a n-state deterministic automaton has a synchronizing word, then it has a synchronizing word of length at most $(n-1)^2$. Berlinkov recently made a breakthrough in the probabilistic analysis of synchronization: he proved that, for the uniform distribution on deterministic automata with n states, an automaton admits a synchronizing word with high probability. In this article, we are interested in the typical length of the smallest synchronizing word, when such a word exists: we prove that a random automaton admits a synchronizing word of length $\mathcal{O}(n \log^3 n)$ with high probability. As a consequence, this proves that most automata satisfy the Černý conjecture.

1998 ACM Subject Classification G.2.1 Combinatorics

Keywords and phrases random automata, synchronization, the Černý conjecture

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.43

1 Introduction

A *synchronizing word* (or a *reset word*) for an automaton is a word that brings that automaton into one and the same state, regardless of the starting position. This notion, first formalized by Černý in the sixties, arises naturally in automata theory and its extensions, and plays an important role in several application areas [17]. Perhaps one of the reasons synchronizing automata are still intensively studied in theoretical computer science is the following question asked by Černý [16] back in 1964: "*Does every synchronizing n-state automaton admits a synchronizing word of length at most $(n-1)^2$?*" The upper bound of $(n-1)^2$, as shown by Černý, is best possible. This question, known as *the Černý conjecture*, is now one of the most famous conjectures in automata theory. Though established for important subclasses of automata, the Černý conjecture remains open in the general case. The best known upper bound, established in the early eighties [13, 6], is $\frac{1}{6}(n^3-n)$. We refer the interested reader to Volkov's article [17] for a more detailed account on the Černý conjecture.

1.1 The probabilistic Cerný conjecture

In this article, we consider the Černý conjecture from a probabilistic point of view (as proposed, for example, by Cameron[1] in [4]). This leads to the following questions, for the uniform distribution on deterministic automata with n states, on a fixed alphabet:

[*] This work is supported by the French National Agency (ANR) through ANR-JCJC-12-JS02-012-01.
[1] Cameron studied the transformation monoid generated by a fixed number of mappings from a set Ω of size n to itself. This is the same as a deterministic automaton, where each mapping correspond to the action of a letter on the set of states Ω. In these settings, "Is the automaton synchronizing?" translates directly into "Does the monoid contain a constant mapping?".

Question 1: Is a random automaton synchronizing *with high probability*?
Question 2: Does a synchronizing automaton admits a synchronizing word of length at most $(n-1)^2$ *with high probability*?

Where *with high probability* means "with probability that tends to 1 as n goes to infinity".

1.2 Main related results

Berlinkov recently made a breakthrough [2] by giving a positive answer to Question 1: he proved that the probability that a random automaton is not synchronizing is $\mathcal{O}(n^{-\frac{1}{2}|A|})$, for an alphabet A with at least two letters.

Question 2 only received partial results so far: Skvortsov and Zaks [15] gave a positive answer for alphabets whose cardinality grows as n^β for $\beta > \frac{1}{2}$. They also proved that the probability of having a short reset word is non-negligible for alphabets with at least four letters [18].

Question 2 can also be simulated and experimental evidence suggests that most automata are synchronized by a short synchronizing word, of length sublinear in the number of states. Note that simulating the second question is nontrivial, as most problems related to the shortest reset word are hard [12] (for instance, deciding whether the shortest reset word as length ℓ is DP-complete, where DP is the closure of NP$\cup$coNP for finite intersections); the best experimental results we are aware of were obtained by Kisielewicz, Kowalski, and Szykula [9]. According to these results, the expected length of the shortest reset word, when it exists, seems to grow in $\Theta(\sqrt{n})$.

Note finally that Berlinkov and Szykula [3] recently used the results of this paper to establish a bound of $n^{3/2+o(1)}$ for the expected value of the shortest reset word in a random synchronizing automaton.

1.3 Our results

In this paper we give a positive answer to Question 2 when the automaton is chosen uniformly among deterministic and complete n-state automata on an alphabet with at least two letters. More precisely, we show that with high probability, a random n-state automaton admits a synchronizing word of length $\mathcal{O}(n \log^3 n)$.

Even if the Černý conjecture is settled in the positive, our main result remains interesting, as it yields that most automata admit a synchronizing word of length almost linear.

Our proof also gives another way to show that automata are synchronizing with high probability, based a method that differ completely from Berlinkov's work. He used recent results on synchronization, as well as some advanced properties of random mappings. In our proof, we directly build words that iteratively shrink the set of states, using only basic discrete probabilities and variations on the probabilistic pigeonhole principle (also known as the Birthday Paradox). The proof proposed by Berlinkov is arguably more complicated, but also more precise, since it gives a sharp estimation of the probability of not being synchronizing[2].

Due to lack of space, the proofs are omitted in this extended abstract.

[2] Knowing the probability of not being synchronizing is important in many situations, especially for the average case analysis of algorithms, as illustrated in the conclusions of [2]. Berlinkov also replies precisely to a question asked by Cameron [4].

2 Definitions and notations

For any integer $n \geq 1$, let $[n] = \{1, \dots, n\}$ be the set of integers between 1 and n. The cardinality of a finite set E is denoted by $|E|$.

2.1 Automata

Let A be a finite alphabet, a *deterministic automaton* on A is a pair (Q, δ), where Q is a finite set of *states* and δ is the *transition function*, a (possibly partial) function from $Q \times A$ to Q. If $p, q \in Q$ and $a \in A$ are such that $\delta(p, a) = q$, then (p, a, q) is the *transition* from p to q labelled by a, and is denoted by $p \xrightarrow{a} q$. It is the a-*transition* outgoing from p. Since we only consider deterministic automata in this article, we simply call them *automata* in the sequel.

An automaton $\mathcal{A} = (Q, \delta)$ on A is classically seen as a labelled directed graph, whose set of vertices is Q and whose edges are the transitions of $\mathcal{A}$.

An automaton is *complete* when its transition function is a total function and *incomplete* otherwise. The transition function is extended inductively to $Q \times A^*$ by setting $\delta(p, \varepsilon) = p$ for every $p \in Q$ and, for every $u \in A^*$, $\delta(p, ua) = \delta(\delta(p, u), a)$ when everything is defined, and undefined otherwise. If $u \in A^*$, we denote by δ_u the (possibly partial) function from Q to Q defined by $\delta_u(p) = \delta(p, u)$, for all $p \in Q$.

If $\mathcal{A} = (Q, \delta)$ is an automaton on A, an *extension* of $\mathcal{A}$ is an automaton $\mathcal{B} = (Q, \lambda)$ on A such that for all $p \in Q$ and for all $a \in A$, if $\delta(p, a)$ is defined then $\lambda(p, a) = \delta(p, a)$. The automaton $\mathcal{B}$ is therefore obtained from $\mathcal{A}$ by adding some transitions. We denote by $\mathbf{Ext}(\mathcal{A})$ the set of all the extensions of an automaton $\mathcal{A}$. If $\mathcal{H}$ is a set of automata, we denote by $\mathbf{Ext}(\mathcal{H})$ the union of all the $\mathbf{Ext}(\mathcal{A})$ for $\mathcal{A} \in \mathcal{H}$.

2.2 Synchronization

Let $\mathcal{A}$ be an automaton on A. Two states p and q of $\mathcal{A}$ are *synchronized* by the word $w \in A^*$ when both $\delta_w(p)$ and $\delta_w(q)$ are defined and equal.

A *synchronizing word* for an automaton $\mathcal{A} = (Q, \delta)$ is a word $w \in A^*$ such that δ_w is a constant map: there exists a state $r \in Q$ such that for every p in Q, $\delta_w(p) = r$. An automaton that admits a synchronizing word is said to be *synchronizing*.

2.3 Mappings

A *mapping* on a set E is a total function from E to E. When E is finite, a mapping f on E can be seen as a directed graph with an edge $i \to j$ whenever $f(i) = j$. An example of such a graph is depicted in Figure 1.

Let f be a mapping on E. The element $x \in E$ is a *cyclic point*[3] of f when there exists an integer $i > 0$ such that $f^i(x) = x$. In the sequel, E will often be the set of states of an automaton, and we will therefore use the term "state" instead of "point".

If f is a mapping on E and $x \in E$, the *height* of x is the smallest $i \geq 0$ such that $f^i(x)$ is a cyclic point. The height of a cyclic point is therefore 0. The *height* of a mapping on E is the maximal height of an element of E. The mapping depicted in Figure 1 has height 3, and the maximal height is reached by the state 9.

[3] We will also say a f-*cyclic point* when the mapping under consideration is not clear in the context.

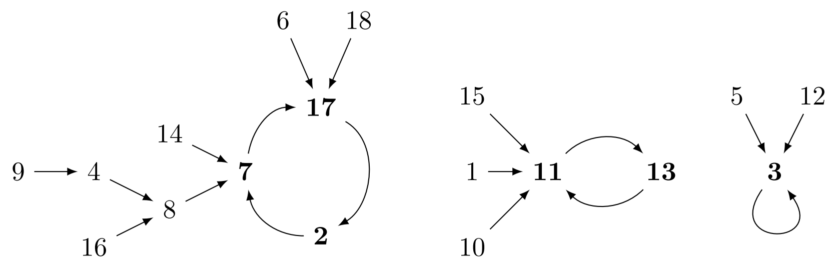


Figure 1 The graph of a mapping for $n = 18$. The cyclic points are indicated in bold.

2.4 Probabilities

Let (E, s) be a pair where E is a set and s is a *size function* from E to $\mathbb{Z}_{\geq 0}$. The pair (E, s) is a combinatorial set[4] when for every integer $n \geq 0$, the set E_n of size-n elements of E is finite. To simplify the definitions, we also assume that $E_n \neq \emptyset$ for every $n \geq 1$, which will always be the case in the following. Let $(\mathbb{P}_n)_{n\geq 1}$ be a sequence of total functions such that for each $n \geq 1$, $\mathbb{P}_n$ is a probability on E_n. We say that a property P holds *with high probability* for $(\mathbb{P}_n)_{n\geq 1}$ when $\mathbb{P}_n[P \text{ holds}] \to 1$ as $n \to \infty$.

We will often consider the *uniform distribution* on E, which is the sequence $(\mathbb{P}_n)_{n\geq 1}$ defined by $\mathbb{P}_n[\{e\}] = \frac{1}{|E_n|}$ for any e in E_n: A sentence like "property P holds with high probability for the uniform distribution on E" therefore means that the probability that P holds tends to 1 as n tends to infinity, when for each n we consider the uniform distribution on E_n. The reader is referred to [5] for more information on combinatorial probabilistic models.

2.5 Random mappings and random p-mappings

A *random mapping* of size $n \geq 1$ is a mapping on $[n]$ taken with the uniform distribution. If p is a probability mass function on $[n]$, a random p-mapping is the distribution on the mappings on $[n]$ such that the probability of a mapping f is $\prod_{i\in[n]} p(f(i))$: the image of each $i \in [n]$ is chosen independently of the others, following the probability p.

A result stated as "a random p-mapping satisfies property P with high probability" means that for *any* sequence $(p_n)_{n\geq 1}$, where p_n is a probability on $[n]$, the probability that a p_n-random mapping on $[n]$ satisfies P tends to 1 as n tends to infinity. It is therefore a strong result that does not depend on the choice of $(p_n)_{n\geq 1}$.

2.6 Random automata

In the sequel, the set of states of an n-state automaton will always be $[n]$. With this condition, there are exactly $n^{|A|n}$ complete automata with n states on $|A|$. For the uniform distribution, each size-n complete automaton has therefore probability $n^{-|A|n}$. See [11] for a recent account on the typical properties of uniform random deterministic automata.

Remark that one can also see this distribution as drawing uniformly at random and independently in $[n]$ the image of each $\delta(p, a)$, for all $p \in [n]$ and $a \in A$. These alternative way to look at random automata will be widely used in the sequel, especially in the following way: Let $\mathcal{A}$ be a fixed incomplete automaton with n states. The uniform distribution on complete

[4] The size is often clear in the context (number of nodes in a tree, ...) and can be omitted.

automata of $\mathbf{Ext}(\mathcal{A})$ is obtained by choosing uniformly at random and independently in $[n]$ the transitions that are undefined in $\mathcal{A}$.

3 Preliminary classical results

In this section, we recall some classical results that will be useful in sequel. Though elementary, these results are the main ingredients of this article.

We start with the following property for synchronizing automata: an automaton is synchronizing if and only if every pair of states can be synchronized.

▶ **Lemma 1.** *Let $\mathcal{A}$ be an n-state automaton and ℓ be a non-negative integer. If for every pair of states (p, q) in $\mathcal{A}$ there exists a word u of length at most ℓ such that $\delta_u(p) = \delta_u(q)$, then $\mathcal{A}$ admits a synchronizing word of length at most $\ell(n-1)$.*

Random mappings and random p-mappings have been studied intensively in the literature [7, 14, 10], using probabilistic techniques or methods from analytic combinatorics. In this section, we only recall basic properties of the typical number of cyclic points and of the typical height of a random p-mapping. This can be achieved using variations on the probabilistic pigeonhole principle only; more advanced techniques can be used to obtain more precise statements[5], but we will only need the following results in the sequel.

▶ **Lemma 2.** *The probability that a random p-mapping of size n has more than $2\sqrt{n \log n}$ cyclic points or that it has height greater than $2\sqrt{n \log n}$ is $\mathcal{O}(\frac{1}{n})$.*

The proof of Lemma 2 consists of two steps. It is first established for uniform random mappings then extended to general p-random mappings, by proving that the uniform case is the worst possible distribution for a p-random mapping.

4 Main Result

The main result of this article is the following theorem.

▶ **Theorem 3.** *Let A be an alphabet with at least two letters. For the uniform distribution, an n-state deterministic and complete automaton on A admits a synchronizing word of length $\mathcal{O}(n \log^3 n)$ with high probability. More precisely, the probability that no such word exists is $\mathcal{O}(n^{-\frac{1}{8}} \log^4 n)$.*

The statement does not hold for alphabets with only one letter, since there are cycles of length greater than 1 in a random mapping with high probability [14]: two distinct states in such a cycle cannot be synchronized.

As a consequence of Theorem 3, a random deterministic and complete automaton is synchronizing with high probability; our proof therefore constitutes an alternative proof of [2] for that property. Our statement is weaker, since Berlinkov also obtained the upper bound $\mathcal{O}(n^{-\frac{1}{2}|A|})$ for the error term (the number of automata that are not synchronizing), which is tight for two-letter alphabets. On the other hand, it is arguably more elementary as we mostly rely on Lemma 2 and some basic discrete probabilities; in any cases, we hope our proof sheds a new light on the reasons why automata are often synchronizing.

[5] For instance, limit distributions of some parameters [5] or even a notion of continuous limit for random mappings [1].

If we consider the uniform distribution on synchronizing automata, we directly obtain that there exists a small synchronizing word with high probability, yielding the following corollary.

▶ **Corollary 4.** *For the uniform distribution on synchronizing deterministic and complete automata on an alphabet with at least two letters, the Černý conjecture holds true with high probability.*

We prove Theorem 3 in two main steps:

- We first construct a word $w_n \in \{a,b\}^*$ such that the image of δ_{w_n} for a random n-state automaton has size at most $n^{1/8}\log^{7/8} n$ with high probability. This is done by building a set $\mathcal{G}_n$ of incomplete automata that have this property, and showing that a random n-state automaton extends an element of $\mathcal{G}_n$ with high probability. Roughly speaking, $\mathcal{G}_n$ and w_n are built by three consecutive applications of Lemma 2, starting with incomplete automata with only a-transitions, which we then augment by b-transitions in two rounds.
- It remains to synchronize those $n^{1/8}\log^{7/8} n$ states. This is done by showing that for a random automaton that extends an element of $\mathcal{G}_n$, with high probability any two of those states can be synchronized by a word of the form $b^i w_n$, with $i \le n^{1/4}$. Lemma 1 is then used to combine these words, and also w_n, into a synchronizing word for the automaton.

The remainder of this section is devoted to a more detailed proof of Theorem 3. For the presentation, we will follow an idea used by Karp in his article on random direct graphs [8]: we start from an automaton with no transition, then add new random transitions during at each step of the construction, progressively improving the synchronization.

Since it is clearly sufficient to establish the result for a two-letter alphabet, we consider that $A = \{a,b\}$ from now on, except for the informal discussion at the beginning of Section 4.3.

4.1 Generating the a-transitions

The first step consists in generating all the a-transitions. This forms a mapping for δ_a that follows the uniformly distribution on size-n mappings. We can therefore apply Lemma 2, and obtain that words of the form a^i can already be used to reduce significantly the number of states to be synchronized.

Let $\alpha_n = \lfloor 2\sqrt{n\log n}\rfloor$ and let $\mathcal{E}_n$ denote the set of incomplete automata $\mathcal{A}$ with n states such that:

1. The defined transitions of $\mathcal{A}$ are exactly its a-transitions.
2. The action δ_a of a has at most α_n cyclic states.
3. The height of δ_a is at most α_n.

▶ **Example 5.** Let $\mathcal{A}$ be an automaton with 18 states, which has only a-transitions and such that δ_a is the mapping of Figure 1 page 4. Its set $\mathbf{Cyc_a}(\mathcal{A})$ is $\{\mathbf{2},\mathbf{3},\mathbf{7},\mathbf{11},\mathbf{13},\mathbf{17}\}$. Since $\alpha_{18} = 14$, the word $u_{18} = a^{14}$ is used to start the synchronization:

$$\{6,7,9,18\} \xrightarrow{u_{18}} \mathbf{2}; \qquad \{3,5,12\} \xrightarrow{u_{18}} \mathbf{3}; \qquad \{4,16,17\} \xrightarrow{u_{18}} \mathbf{7};$$
$$\{11\} \xrightarrow{u_{18}} \mathbf{11}; \qquad \{1,10,13,15\} \xrightarrow{u_{18}} \mathbf{13}; \qquad \{2,8,14\} \xrightarrow{u_{18}} \mathbf{17}.$$

As there are $6 \le \alpha_{18}$ cyclic states and since this mapping's height is $3 \le \alpha_{18}$, the automaton $\mathcal{A}$ is an element of $\mathcal{E}_{18}$.

As the action of the letter a in a uniform random complete automaton is exactly a uniform random mapping, the following result is a direct consequence of Lemma 2.

▶ **Lemma 6.** *A random complete automaton with n states extends an element of $\mathcal{E}_n$ with high probability. More precisely, the probability that such an automaton does not extend an element of $\mathcal{E}_n$ is $\mathcal{O}(\frac{1}{n})$.*

For any automaton $\mathcal{A}$ whose a-transitions are all defined, let $\mathbf{Cyc_a}(\mathcal{A})$ denote its set of δ_a-cyclic states. They also are the δ_a-cyclic states of any automaton that extends $\mathcal{A}$.

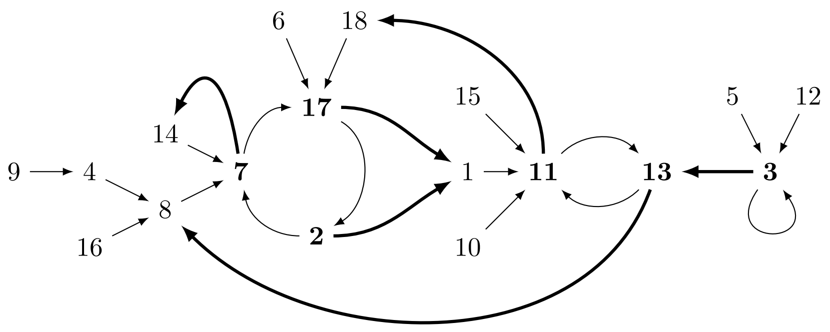
Let $u_n = a^{\alpha_n}$. By Lemma 6, we can already start the synchronization using u_n, as the image of the set of states $[n]$ by δ_{u_n} is included in $\mathbf{Cyc_a}(\mathcal{A})$, which is much smaller than n with high probability. In the sequel, we therefore work on synchronizing the elements of $\mathbf{Cyc_a}(\mathcal{A})$.

4.2 Adding some random b-transitions

Let $\mathcal{A}$ be a fixed element of $\mathcal{E}_n$. We are now working on $\mathbf{Ext}(\mathcal{A})$ and we consider the process of adding a random b-transition starting from every state of $\mathbf{Cyc_a}(\mathcal{A})$.

Let $\mathcal{B} \in \mathbf{Ext}(\mathcal{A})$ be an automaton obtained this way and let $f_\mathcal{B}$ denote the restriction of δ_{bu_n} to $\mathbf{Cyc_a}(\mathcal{A})$. It is a total map, since all the needed b-transitions are defined. Moreover, the image of $f_\mathcal{B}$ is included in $\mathbf{Cyc_a}(\mathcal{A})$, as $f_\mathcal{B}(x) = \delta_{bu_n}(x) = \delta_{u_n}(\delta_b(x))$, for every $x \in \mathbf{Cyc_a}(\mathcal{A})$. Hence $f_\mathcal{B}$ is a total map from $\mathbf{Cyc_a}(\mathcal{A})$ to itself.

▶ **Example 7.** This is the automaton of Example 5, where the b-transitions originating from the elements of $\mathbf{Cyc_a}(\mathcal{A})$ have been added (in bold):



The map $f_\mathcal{B}$, which is the restriction of δ_{bu_n} to $\mathbf{Cyc_a}(\mathcal{A})$, is depicted below. An edge $\mathbf{p} = x \Longrightarrow \mathbf{q}$ means that $\delta_b(p) = x$ and $\delta_{u_n}(x) = q$, so that $f_\mathcal{B}(p) = q$:

$$\begin{tikzcd}
 & & & \mathbf{3} \arrow[dl, Rightarrow, "13"'] \\
\mathbf{7} \arrow[r, Rightarrow, "14"] & \mathbf{17} \arrow[r, Rightarrow, bend left, "1"] & \mathbf{13} \arrow[l, Rightarrow, bend left, "8"] & \\
 & & & \mathbf{2} \arrow[ul, Rightarrow, "1"] & \mathbf{11} \arrow[l, Rightarrow, "18"']
\end{tikzcd}$$

From a probabilistic point of view, if we fix $\mathcal{A}$ and build $\mathcal{B}$ by adding uniformly at random and independently the b-transitions that start from the states of $\mathbf{Cyc_a}(\mathcal{A})$, the induced distribution for the mapping $f_\mathcal{B}$ is usually not the uniform distribution on the mappings of $\mathbf{Cyc_a}(\mathcal{A})$. More precisely, for any $q \in \mathbf{Cyc_a}(\mathcal{A})$ the probability that the image by $f_\mathcal{B}$ of an element of $\mathbf{Cyc_a}(\mathcal{A})$ is q is proportional to the number of preimages of q by δ_{u_n}. It is exactly

$\frac{1}{n}|\delta_{u_n}^{-1}(\{q\})|$, the probability that a random state is mapped to q when reading u_n. For any word $\omega \in A^*$, let $\mathbb{P}_{\mathcal{A},\omega}$ be the function from $[n]$ to $[0,1]$ defined by

$$\mathbb{P}_{\mathcal{A},\omega}(q) = \frac{|\delta_\omega^{-1}(\{q\})|}{n}, \text{ for all } q \in [n]. \tag{1}$$

From the observations above, we get that once $\mathcal{A}$ is fixed, $f_\mathcal{B}$ is a random p-mapping, where the distribution on $\mathbf{Cyc_a}(\mathcal{A})$ is given by the restriction of $\mathbb{P}_{\mathcal{A},u_n}$ to $\mathbf{Cyc_a}(\mathcal{A})$.

Let $\beta_n = \lfloor 3\, n^{1/4} \log^{3/4} n \rfloor$. Applying Lemma 2 to $f_\mathcal{B}$ yields the following result.

▶ **Lemma 8.** *Let $\mathcal{A}$ be a fixed automaton of $\mathcal{E}_n$. Consider the random process of building $\mathcal{B}$ by adding a b-transition to every element of* $\mathbf{Cyc_a}(\mathcal{A})$*, choosing the target uniformly and independently in $[n]$. For n sufficiently large, the probability that $f_\mathcal{B}$ has more than β_n cyclic states or that it has height greater than β_n is smaller than $\frac{M}{n^{1/4}}$, for some positive constant M that does not depends on $\mathcal{A}$.*

For any automaton $\mathcal{B}$ whose a-transitions are all defined and whose b-transitions starting from an element of $\mathbf{Cyc_a}(\mathcal{B})$ are also all defined, let $\mathbf{Cyc_f}(\mathcal{B})$ denote the set of $f_\mathcal{B}$-cyclic states of $\mathcal{B}$.

Let $v_n = u_n(bu_n)^{\beta_n}$. At this point, the number of states to be synchronized has been reduced to less than β_n with high probability, since the image of δ_{v_n} is usually included in $\mathbf{Cyc_f}(\mathcal{B})$. It has been achieved by generating all the a-transitions, but using only the b-transitions that start from the δ_a-cyclic states: there still remain at least $n - \alpha_n$ undefined b-transitions that can be used to continue the synchronization. Nonetheless, before going on, we first refine the construction of $\mathcal{B}$ introduced in this section by forbidding some cases, for technical reasons explained in the next section.

4.3 Forbidding correlated shapes

The number of states to be synchronized has been reduced to no more than β_n states with high probability, but this quantity is still too large. For the technique used at the end of the proof, we need to shrink this set once more. Should the alphabet contain one more letter c, we could use the same kind of construction as in Section 4.2, and be left with at most, roughly, $n^{1/8}$ states to synchronize. This is because c-transitions can be generated independently of what has been done during the previous steps.

Some care is required to adapt this idea for a two-letter alphabet. We aim at using the word bb instead of the letter c in the informal description above. Let $\mathcal{B}$ be an incomplete automaton that extends $\mathcal{A} \in \mathcal{E}_n$ and whose defined transitions are all the a-transitions and also the b-transitions that start from the δ_a-cyclic states. We are interested in building an automaton $\mathcal{C}$ from $\mathcal{B}$, by adding some new random b-transitions, in a way such that δ_{bbv_n} is totally defined on $\mathbf{Cyc_f}(\mathcal{B})$. It means that for every $q \in \mathbf{Cyc_f}(\mathcal{B})$, the state $\delta_b(q)$ must have an outgoing b-transition in $\mathcal{C}$. For such an extension $\mathcal{C}$ of $\mathcal{B}$, let $g_\mathcal{C}$ denote the restriction of δ_{bbv_n} to $\mathbf{Cyc_f}(\mathcal{B})$.

The main point here is that for a fixed $\mathcal{B}$, we want $g_\mathcal{C}$ to be defined as a random p-mapping, so that we can use Lemma 2 once more. There are, *a priori*, two kind of issues that can prevent this from happening:

1. When there exists a state $q \in \mathbf{Cyc_f}(\mathcal{B})$ such that the b-transition starting from $\delta_b(q)$ is already defined in $\mathcal{B}$, that is, when $\delta_b(q) \in \mathbf{Cyc_a}(\mathcal{B})$.
2. When two distinct states q and q' in $\mathbf{Cyc_f}(\mathcal{B})$ are such that $\delta_b(q) = \delta_b(q')$.

Fortunately, the second case cannot occur: if $\delta_b(q) = \delta_b(q')$ then $f_\mathcal{B}(q) = f_\mathcal{B}(q')$, which is not possible for two distinct $f_\mathcal{B}$-cyclic states.

The first case can occur, and then the image of $\delta_b(q)$ by b is already defined in $\mathcal{B}$ and therefore $g_{\mathcal{C}}$ does not follow a p-distribution when we build $\mathcal{C}$ by generating the missing transitions uniformly at random[6].

Conversely, if for every $q \in \mathbf{Cyc_f}(\mathcal{B})$, $\delta_b(q) \notin \mathbf{Cyc_a}(\mathcal{B})$, then it is easy to verify that $g_{\mathcal{C}}$ is a random p-mapping: the image of $q \in \mathbf{Cyc_f}(\mathcal{B})$ by $g_{\mathcal{C}}$ is a given x when $\delta_{bbv_n}(q) = x$, which is equivalent to $\delta_b(\delta_b(q)) \in \delta_{v_n}^{-1}(\{x\})$. Since $\delta_b(\delta_b(q))$ is chosen uniformly at random in $[n]$, it happens with probability $\mathbb{P}_{\mathcal{B},v_n}(x)$, using the notation of Equation (1).

We therefore forbid the bad cases and define the set $\mathcal{F}_n$ of incomplete automata $\mathcal{B}$ with n states such that (we add the last condition to what was done in the previous section):

1. $\mathcal{B}$ extends an element of $\mathcal{E}_n$.
2. The defined transitions of $\mathcal{B}$ are all the a-transitions and the b-transitions starting from the states of $\mathbf{Cyc_a}(\mathcal{B})$.
3. The map $f_{\mathcal{B}}$ has height at most β_n and has at most β_n cyclic states.
4. For every $q \in \mathbf{Cyc_f}(\mathcal{B})$, $\delta_b(q) \notin \mathbf{Cyc_a}(\mathcal{B})$.

▶ **Example 9.** The automaton of Example 7 is in $\mathcal{F}_n$. For the fourth condition, observe that the $f_{\mathcal{B}}$-cyclic states are **13** and **17**. Their images by δ_b, which are 8 and 1 respectively, are not in $\mathbf{Cyc_a}(\mathcal{B})$. The fact that $\delta_b(\mathbf{3})$ is in $\mathbf{Cyc_a}(\mathcal{B})$ is not a problem here, since **3** is not an $f_{\mathcal{B}}$-cyclic state.

If we forget the last condition in the definition of $\mathcal{F}_n$, the other requirements hold with high probability for every fixed $\mathcal{A} \in \mathcal{E}_n$, as a consequence of Lemma 8. Lemma 10 below states that after our additional restriction, the set we obtain is still sufficiently large.

▶ **Lemma 10.** *With high probability a random complete automaton with n states extends an element of $\mathcal{F}_n$. More precisely, the probability that it does not extend an element of $\mathcal{F}_n$ is at most $n^{-1/4}\log^2 n$, for n sufficiently large.*

4.4 Adding more random b-transitions

Starting from an element of $\mathcal{B} \in \mathcal{F}_n$, we can now use the idea explained at the beginning of Section 4.3, and add the random b-transitions that are needed for δ_{bb} to be totally defined on $\mathbf{Cyc_f}(\mathcal{B})$. For such an extension $\mathcal{C}$ of $\mathcal{B}$, recall that the mapping $g_{\mathcal{C}}$ is the restriction of δ_{bbv_n} to $\mathbf{Cyc_f}(\mathcal{B})$. Let $\mathbf{Cyc_g}(\mathcal{C})$ denote the set of $g_{\mathcal{C}}$-cyclic states in $\mathcal{C}$. Thanks to the last condition of the definition of $\mathcal{F}_n$, we need to randomly choose the b-transitions starting from the images by δ_b of $\mathbf{Cyc_f}(\mathcal{B})$, which are all distinct since two distinct states of $\mathbf{Cyc_f}(\mathcal{B})$ cannot have the same image by δ_b.

Let $\gamma_n = \lfloor 2\,n^{1/8}\log^{7/8} n \rfloor$ and let $X_{\mathcal{B}}$ denote the set of images of $\mathbf{Cyc_f}(\mathcal{B})$ by δ_b, i.e. $X_{\mathcal{B}} = \{\delta_b(x) : x \in \mathbf{Cyc_f}(\mathcal{B})\}$. We define the set $\mathcal{G}_n$ of incomplete automata $\mathcal{C}$ with n states that satisfy the following conditions:

1. $\mathcal{C}$ extends an automaton $\mathcal{B}$ of $\mathcal{F}_n$.
2. The only b-transitions of $\mathcal{C}$ are those starting from $\mathbf{Cyc_a}(\mathcal{B})$ and from $X_{\mathcal{B}}$.
3. The map $g_{\mathcal{C}}$ has no more than γ_n cyclic states and has height at most γ_n.
4. For every $q \in \mathbf{Cyc_g}(\mathcal{C})$ the b-transition of $\delta_{bb}(q)$ is undefined.

The last condition in the definition of $\mathcal{G}_n$ is useful for the same kind of reasons than the last condition of $\mathcal{F}_n$ is. It ensures some independence for the final step of the synchronization, which is presented in Section 4.5.

[6] Except in the very degenerate case where the restriction of δ_{bb} to $\mathbf{Cyc_f}(\mathcal{B})$ is already a totally defined and constant map in $\mathcal{B}$.

▶ **Lemma 11.** *With high probability, a random complete automaton with n states extends an element of $\mathcal{G}_n$. More precisely, the probability it does not extends an element of $\mathcal{G}_n$ is $\mathcal{O}(\frac{1}{\gamma_n})$.*

Let $w_n = v_n(bbv_n)^{\gamma_n}$. Lemma 11 ensures that for a random complete automaton $\mathcal{A}$, the image of δ_{w_n} is usually included in $\mathbf{Cyc_g}(\mathcal{A})$, which has size at most γ_n. This concludes the first part of the synchronization: with high probability, the word w_n maps the set of states of $\mathcal{A}$ to the much smaller set of states $\mathbf{Cyc_g}(\mathcal{A})$.

4.5 Synchronizing the remaining states

Let $\lambda_n = \lfloor n^{1/4} \rfloor$ and let $\mathcal{C}$ be a fixed automaton of $\mathcal{G}_n$. Starting from $\mathcal{C} \in \mathcal{G}_n$, we now prove that the elements of $\mathbf{Cyc_g}(\mathcal{C})$ can be synchronized with high probability when setting randomly the undefined b-transitions. We follow the idea given at the beginning of Section 4 and first prove that with high enough probability, two states of $\mathbf{Cyc_g}(\mathcal{C})$ can be synchronized by a word of the form $b^j w_n$, for some integer $j \geq 0$.

Let q and r be two states of $\mathbf{Cyc_g}(\mathcal{C})$. By definition of $\mathcal{G}_n$, the states $q_2 = \delta_{bb}(q)$ and $r_2 = \delta_{bb}(r)$ have no outgoing b-transitions in $\mathcal{C}$. For $i \geq 3$, we iteratively build a sequence of uniform and independent pairs (q_i, r_i) of $[n] \times [n]$, and set $\delta_b(q_{i-1}) = q_i$ and $\delta_b(r_{i-1}) = r_i$. We stop this random process if at any step either $\delta_{w_n}(q_i) = \delta_{w_n}(r_i)$, or q_i (or r_i) already has a defined b-transition, or $i = \lambda_n$. By studying the probability that this random process halts because of the condition $\delta_{w_n}(q_i) = \delta_{w_n}(r_i)$, we obtain the following Lemma.

▶ **Lemma 12.** *Let $\mathcal{C} \in \mathcal{G}_n$ and let q and r be two distinct states of $\mathbf{Cyc_g}(\mathcal{C})$. If we add all the missing b-transitions to $\mathcal{C}$ by drawing them uniformly at random and independently, then the probability that for all $j \in \{0, \ldots, \lambda_n\}$ we have $\delta_{b^j \cdot w_n}(r) \neq \delta_{b^j \cdot w_n}(r)$ is at most $n^{-3/8} \log^2 n$, for n sufficiently large.*

To conclude the proof of Theorem 3, we use the union bound: for any automaton $\mathcal{A}$ that extends an element of $\mathcal{G}_n$, which happens with high probability, there are less than γ_n^2 pairs of states in $\mathbf{Cyc_g}(\mathcal{A})$; the probability that one of these pairs (q, r) cannot be synchronized using a word of the form $b^j \cdot w_n$ is therefore at most $\gamma_n^2 \cdot n^{-3/8} \log^2 n$, which is $\mathcal{O}(n^{-\frac{1}{8}} \log^4 n)$.

To obtain the length of the synchronizing word, we apply Lemma 1 to the elements of $\mathbf{Cyc_g}(\mathcal{A})$: with high probability there are at most γ_n such states, which can be pairwise synchronized using words of the form $b^j w_n$, of length at most $|w_n| + \lambda_n$. Hence, the set $\mathbf{Cyc_g}(\mathcal{A})$ can be synchronized using a word z of length at most $(\gamma_n - 1)(|w_n| + \lambda_n)$, which is asymptotically equivalent to $n \log^3 n$. This conclude the proof, as $w_n z$ is synchronizing and has length which is also asymptotically equivalent to $n \log^3 n$.

5 Conclusion

In this article we proved that most complete automata are synchronizing and admit a synchronizing word of length $\mathcal{O}(n \log^3 n)$.

Our proof can be turned into an heuristic that try to find a short synchronizing word, which succeeds with high probability for uniform random automata: δ_{w_n} and $\mathbf{Cyc_g}(\mathcal{A})$ can be computed by just verifying some conditions on the height and cycle length of three mappings; once it is done, checking whether the property of Lemma 12 holds for every pair of elements of the image of δ_{w_n} can be achieved in sublinear time, as it is very small with high probability. Experiments seem to indicate that this algorithm behaves better in practice than its theoretical analysis: it looks like an important proportion of automata that fail to fulfill every step of our construction are still detected as synchronizing by the combination of computing δ_{w_n} and synchronizing the states of its image with the b^j's.

A natural continuation of this work is to prove that with high probability automata are synchronized by words that are way shorter than $n \log^3 n$. Experiments have been done [9], and seem to indicate that the expected length of the smallest synchronizing word is often sublinear, probably in $\Theta(\sqrt{n})$. There is plenty of room to improve our construction, as the synchronizing words we obtain have very specific shapes. It still might be quite difficult to match the bounds predicted in [9].

Acknowledgments. The author would like to thank Marie-Pierre Béal and Dominique Perrin for their interest in this work since the very beginning, and Mikhail Berlinkov for our fruitful discussions on this topic.

References

1 David Aldous, Grégory Miermont, and Jim Pitman. Brownian bridge asymptotics for random p-mappings. *Electron. J. Probab*, 9:37–56, 2004.

2 Mikhail V. Berlinkov. On the probability of being synchronizable. In Sathish Govindarajan and Anil Maheshwari, editors, *Algorithms and Discrete Applied Mathematics – Second International Conference, CALDAM 2016, Thiruvananthapuram, India, February 18-20, 2016, Proceedings*, volume 9602 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2016. doi:10.1007/978-3-319-29221-2_7.

3 Mikhail V. Berlinkov and Marek Szykula. Algebraic synchronization criterion and computing reset words. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Mathematical Foundations of Computer Science 2015 – 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*, volume 9234 of *Lecture Notes in Computer Science*, pages 103–115. Springer, 2015. doi:10.1007/978-3-662-48057-1_8.

4 Peter J Cameron. Dixon's theorem and random synchronization. *Discrete Mathematics*, 313(11):1233–1236, 2013.

5 Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

6 Peter Frankl. An extremal problem for two families of sets. *Eur. J. Comb.*, 3:125–127, 1982.

7 Bernard Harris. Probability distributions related to random mappings. *The Annals of Mathematical Statistics*, 31(4):1045–1062, 1960.

8 Richard M. Karp. The transitive closure of a random digraph. *Random Struct. Algorithms*, 1(1):73–94, 1990. doi:10.1002/rsa.3240010106.

9 Andrzej Kisielewicz, Jakub Kowalski, and Marek Szykula. A fast algorithm finding the shortest reset words. In Ding-Zhu Du and Guochuan Zhang, editors, *COCOON*, volume 7936 of *Lecture Notes in Computer Science*, pages 182–196. Springer, 2013. doi:10.1007/978-3-642-38768-5_18.

10 Valentin F. Kolčin. *Random Mappings: Translation Series in Mathematics and Engineering*. Translations series in mathematics and engineering. Springer London, Limited, 1986.

11 Cyril Nicaud. Random deterministic automata. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 5–23. Springer, 2014. doi:10.1007/978-3-662-44522-8_2.

12 Jörg Olschewski and Michael Ummels. The complexity of finding reset words in finite automata. In Petr Hlinený and Antonín Kucera, editors, *MFCS*, volume 6281 of *Lecture Notes*

in Computer Science, pages 568–579. Springer, 2010. doi:10.1007/978-3-642-15155-2_50.

13 Jean-Eric Pin. On two combinatorial problems arising from automata thery. *Annals of Discrete Mathematics*, 17:535–548, 1983.

14 Jean-Jacques Quisquater and Joos Vandewalle, editors. *Advances in Cryptology – EUROCRYPT'89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings*, volume 434 of *Lecture Notes in Computer Science*. Springer, 1990.

15 Evgeny S. Skvortsov and Yulia Zaks. Synchronizing random automata. *Discrete Mathematics & Theoretical Computer Science*, 12(4):95–108, 2010.

16 J. Černý. Poznámka k. homogénnym experimentom s konecnymi automatmi. *Matematicko-fyzikalny Časopis Slovensk*, 14, 1964.

17 Mikhail V. Volkov. Synchronizing Automata and the Cerny Conjecture. In Carlos Martín-Vide, Friedrich Otto, and Henning Fernau, editors, *Language and Automata Theory and Applications, Second International Conference, LATA 2008, Tarragona, Spain, March 13-19, 2008. Revised Papers*, volume 5196 of *Lecture Notes in Computer Science*, pages 11–27. Springer, 2008. doi:10.1007/978-3-540-88282-4_4.

18 Yulia Zaks and Evgeny S. Skvortsov. Synchronizing random automata on a 4-letter alphabet. *Journal of Mathematical Sciences*, 192:303–306, 2013.

A Direct-Sum Theorem for Read-Once Branching Programs*

Anup Rao[1] and Makrand Sinha[2]

1 **Computer Science and Engineering, University of Washington, Seattle WA, USA**
`anuprao@cs.washington.edu`
2 **Computer Science and Engineering, University of Washington, Seattle WA, USA**
`makrand@cs.washington.edu`

Abstract

We study a direct-sum question for read-once branching programs. If $M(f)$ denotes the minimum average memory required to compute a function $f(x_1, x_2, \ldots, x_n)$ how much memory is required to compute f on k independent inputs that arrive in parallel? We show that when the inputs are sampled independently from some domain $\mathcal{X}$ and $M(f) = \Omega(n)$, then computing the value of f on k streams requires average memory at least $\Omega\left(k \cdot \frac{M(f)}{n}\right)$.

Our results are obtained by defining new ways to measure the information complexity of read-once branching programs. We define two such measures: the *transitional* and *cumulative* information content. We prove that any read-once branching program with transitional information content $\mathtt{I}$ can be simulated using average memory $\mathcal{O}(n(\mathtt{I}+1))$. On the other hand, if every read-once branching program with cumulative information content $\mathtt{I}$ can be simulated with average memory $\mathcal{O}(\mathtt{I}+1)$, then computing f on k inputs requires average memory at least $\Omega(k \cdot (M(f) - 1))$.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation

Keywords and phrases Direct-sum, Information complexity, Streaming Algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.44

1 Introduction

In this paper we investigate direct-sum questions for read-once branching programs (equivalently, streaming algorithms). Recall that an input to a read-once branching program is a sequence of n updates $x_1, \ldots, x_n$ arriving sequentially in time, and the branching program at the end must compute a function $f(x_1, x_2, \ldots, x_n)$. The complexity measure of interest is the amount of memory that is needed to carry out the computation. Here the memory used by the program at time t is the logarithm of the number of potential states that the program can be in after reading the inputs $x_1, \ldots, x_t$.

We are interested in how the complexity of a problem changes when the branching program must process k independent inputs that arrive in parallel. The program now gets k inputs $x^1 = x^1_1, \ldots, x^1_n, x^2 = x^2_1, \ldots, x^2_n, \ldots, x^k = x^k_1, \ldots, x^k_n$, where the inputs $x^1_t, \ldots, x^k_t$ arrive simultaneously in the t'th time-step. Obviously one can process each of the inputs independently, giving a branching program that uses k times as much memory. The central

* Supported by the National Science Foundation under agreement CCF-1016565, an NSF Career award, and by the Binational Science Foundation under agreement 2010089.

question that we investigate in this paper is: are there interesting functions f for which the best branching program that computes f on k independent inputs does *not* operate independently on each input? This question is dual to another interesting question: When can we effectively reduce the memory of a branching program without compromising its accuracy?

Viewing these read-once branching programs as streaming algorithms, these questions also make a lot of sense in the context of the most common applications for streaming algorithms like internet traffic analysis or data from multiple satellites. They also make sense from a theoretical perspective: they help to identify exactly what makes some streaming tasks hard and others easy.

The extensive literature on branching programs is mostly concerned with understanding the maximum number of bits of memory used by the branching program throughout its run. One can imagine pathological cases one can effectively process k inputs at the same cost as processing a single input using this measure of complexity. Suppose there is a uniformly random block of n/k^3 consecutive updates that contains information in the input, and all other updates are set to 0. Then without loss of generality, the best (read-once) branching program uses almost no memory for most of the time, and some memory to process the block of important inputs. When the program processes k parallel inputs, it is very likely that the k informative blocks will not overlap in time, and so the maximum memory usage remains unchanged. Thus, if we are only aiming for a read-once branching program that succeeds with high probability over this distribution of inputs, one need not increase the memory at all!

However, we see that the *average memory* usage per unit time-step does increase by a factor of k in this last example. The average memory is defined to be the number of bits of memory used on an average time-step. Arguably from the streaming viewpoint, the average memory is what we care about when considering practical applications of streaming algorithms. Another appealing reason to consider average memory as a complexity measure is that some known streaming lower bounds actually yield lower bounds on the average memory. For example, the lower bound proofs for approximating the frequency moments [1, 4, 10, 22] and for approximating the length of the longest increasing subsequence [19] can be easily adapted to give matching lower bounds for average memory. In the rest of this work we focus on the average memory used by the branching program.

Note that it is standard for analyzing branching programs to count the number of states in each layer, but since we will be working with entropy it will be more convenient for us to talk about the memory required to store each layer. As such to present our results we adopt the point of view of inputs as streams and a branching program as a streaming algorithm.

1.1 Related Work

The interest in the field of streaming algorithms was renewed by the seminal paper of Alon, Matias and Szegedy [1] who gave algorithms for approximating lower frequency moments and also showed that lower bounds in the multi-party number-in-hand communication model implied memory lower bounds for streaming algorithms approximating the higher frequency moments. Since then, lower bounds in communication complexity (and more recently in information complexity) have found applications in proving memory lower bounds in the streaming model (see [1, 4, 10, 32, 19, 23, 22, 28] for some of them).

Questions analogous to the ones we study here have been studied in the setting of two-party communication complexity and information complexity [5, 9, 6]. It was shown in [21] that there are communication tasks that can be solved much more efficiently in parallel than by naively solving each one independently.

Combining these results about parallelizing communication with known methods for proving lower bounds on streaming algorithms gives several interesting worst-case memory lower bounds for computing natural functions on k parallel streams. To give an example, it is known that computing $(1+\varepsilon)$ approximation of the p^{th} frequency moment for $p \neq 1$ requires worst-case memory $\Omega(1/\varepsilon^2)$ [32, 22]. Combining this with the results of [9] one can show that computing $(1+\varepsilon)$ approximation of the frequency moment on k streams in parallel requires $\Omega(k/\varepsilon^2)$ memory in the worst-case. We do not give the proof here, since it is relatively straightforward.

A related model is that of *dynamic distributed functional monitoring* introduced by Cormode, Muthukrishnan and Yi [15] where there are multiple sites receiving data streams and communicating with a central coordinator who wants to maintain a function of all the input streams. Recent progress has been made in understanding the communication complexity of various tasks in this model [15, 33, 34]. Variants of this model have been studied extensively in relation to databases and distributed computing (see [13, 14, 31, 30, 12, 16, 2, 27, 26, 3] for some of the applications). Another closely related model is the multi-party *private message passing* model introduced in [18]. Any lower bound proved in the message passing model implies a lower bound in the streaming model. Many works have studied this model and its variants (see [23, 20, 29, 7, 11, 25] for some of them). These works do not appear to have any connection to the questions we study here.

2 Our Results

Our results are proved in the setting of average-case complexity: we assume that there is a known distribution on inputs, and consider the performance of algorithms with respect to that distribution. Let $\mathcal{A}$ be a randomized streaming algorithm which receives an input stream $X_1, \ldots, X_n$ sampled from a distribution $p(x_1, \ldots, x_n)$. Throughout this paper we will only consider the case when p is a product distribution except in Section 4.1, where we discuss the issues that arise when considering non-product input distributions.

Let $M_1, \ldots, M_n$ denote the contents of the memory of the algorithm at each of the time-steps. Let $|M_t|$ denote the number of bits used to store M_t. The average memory used by the algorithm is $(1/n)\sum_{t=1}^{n}|M_t|$. Let $M(f)$ denote the minimum average memory required to compute a function f with probability $2/3$ when the inputs are sampled according to p.

Let $p^k(x)$ denote the product distribution on k independent streams, each identically distributed as $p(x)$, where the resulting streams arrive synchronously in parallel. Thus at time t the input is the t^{th} element of all the k streams. Write f^k to denote the function that computes f on each of the k streams. Then we prove,

▶ **Theorem 2.1.**

$$M(f^k) = \Omega\left(k\left(\frac{M(f)}{n} - 1\right)\right).$$

Theorem 2.1 is proved by a reduction that compresses streaming algorithms with regards to its information complexity. There are several reasonable measures of information complexity for streaming algorithms. Here we define two such information complexity measures. We use Shannon's notion of mutual information, which is defined in the preliminaries (Section 3).

The *transitional information content* captures the average amount of information that the algorithm learns about the next input conditioned on its current state.

▶ **Definition 2.2** (Transitional Information). $\mathsf{IC}^{tr}(\mathcal{A}) = \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t | M_{t-1})$.

The *cumulative information content* measures the average amount of information that the streaming algorithm remembers about the inputs seen so far.

▶ **Definition 2.3** (Cumulative Information). $\mathsf{IC}^{cum}(\mathcal{A}) = \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_1 \dots X_t)$.

Note that both the transitional and the cumulative information content for an algorithm are bounded by the average memory used by the algorithm. We prove that algorithms with low transitional information can be efficiently simulated:

▶ **Theorem 2.4.** *Every streaming algorithm with transitional information content* $\mathtt{I}$ *can be simulated with average memory* $\mathcal{O}(n\mathtt{I} + n)$.

The above theorem is tight as the following example shows. Let the input x be sampled from the uniform distribution on $\{0,1\}^n$ (*i.e.* each update x_i for $i \in [n]$ is a bit). Consider the streaming algorithm $\mathcal{A}$ which remembers all the updates seen so far and outputs $x_1, \dots, x_n$ at the end. The average memory used by the algorithm is $\Omega(n)$ while the transitional information content of this algorithm is 1. In this case the compression algorithm given by the above theorem would simulate $\mathcal{A}$ with average memory $O(n)$ which is the best one could hope for.

Finally, we show that if algorithms with low cumulative information can be simulated, then one can obtain no savings when parallelizing streaming algorithms:

▶ **Theorem 2.5.** *If every algorithm with cumulative information* $\mathtt{I}$ *can be simulated using average memory* $\mathcal{O}(\mathtt{I})$*, then* $M(f^k) = \Omega\left(k \cdot (M(f) - 1)\right)$.

In Section 5, we discuss more about the possibility of compressing algorithms with low cumulative information content.

3 Preliminaries

Throughout this report, the base of all logarithms is 2. Random variables will be denoted by capital letters and the values that they attain will be denoted by lower-case letters. Given $x = x_1, \dots, x_n$, we write $x_{\le i}$ to denote the sequence $x_1, x_2, \dots, x_i$. We define $x_{<i}, x_{>i}$ and $x_{\ge i}$ similarly. We write x_{-i} to denote $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$.

We use $p(x)$ to denote both the distribution on the variable x and the probability $\mathbb{P}_p[X = x]$, the distinction will be clear from context. For any joint random variables X and Y, we will write $X|Y = y$ to denote the random variable X conditioned on the event $Y = y$ and use $p(x|y)$ to denote the distribution of $X|Y = y$ as well as the probability $\mathbb{P}_p[X = x|Y = y]$.

We denote by $p^k(x)$ the product distribution sampling k independent copies of x according to p. Given a joint distribution $p(x, y, z)$, we write $p(x, y)$ to denote the marginal distribution (or probability according to the marginal distribution) on the variables x and y. We often write $p(xy)$ instead of $p(x, y)$ to make the notation more concise. When X, Y are random variables, XY denotes the random variable that is the concatenation of X and Y.

Let X, W, M be random variables distributed according to $p(x, w, m)$. We say that they form a Markov chain iff $p(x, w, m) = p(w) \cdot p(x|w) \cdot p(m|w)$ and we denote this by $X - W - M$. In some cases we will have Markov chains where W determines M ($p(m|w)$ is a point distribution). To emphasize this we will write this Markov chain as $X - W \to M$. For brevity we will write $X|R - W|R - M|R$ to assert that $p(xwm|r)$ is a Markov chain for every r.

3.1 Information Theory Basics

Here we collect some standard facts from information theory. For more details, we refer the reader to the textbook [17]. For a discrete random variable X with probability distribution $p(x)$, the entropy of X is defined as

$$H(X) = \mathbb{E}_{p(x)}\left[\log \frac{1}{p(x)}\right].$$

For any two random variables X and Y with the joint distribution $p(x,y)$, the entropy of X conditioned on Y is defined as $H(X|Y) = \mathbb{E}_{p(y)}[H(X|Y=y)]$. The conditional entropy $H(X|Y)$ is at most $H(X)$ where the equality holds if and only if X and Y are independent.

The mutual information between X and Y is defined as $I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$. Similarly, the conditional mutual information $I(X;Y|Z)$ is defined to be $H(X|Z) - H(X|YZ)$. If X and Y are independent then $I(X;Y) = 0$. Moreover, $0 \leq I(X;Y) \leq \min\{H(X), H(Y)\}$. A standard fact about mutual information is the chain rule: For jointly distributed random variables $X_1, \ldots, X_n$, Y and Z,

$$I(X_1, \ldots, X_n; Y|Z) = \sum_{i=1}^{n} I(X_i; Y|X_{<i}Z).$$

▶ **Lemma 3.1.** *If Y and Z are independent, $I(X;Y) \leq I(X;Y|Z)$.*

Proof. We repeatedly use the chain rule:

$$I(X;Y) \leq I(X;Y) + I(Z;Y|X) = I(XZ;Y) = I(Z;Y) + I(X;Y|Z) = I(X;Y|Z). \quad ◀$$

▶ **Proposition 3.2** (Data Processing Inequality). *Let X, W and M be random variables such that $X - W - M$, then $I(X;M) \leq I(X;W)$.*

▶ **Proposition 3.3.** *Let X, Y, Z and W be random variables such that $XY - Z - W$, then $I(X;Y|ZW) = I(X;Y|Z)$.*

Proof. Using the chain rule we expand $I(XW;Y|Z)$ in two different ways:

$$I(W;Y|Z) + I(X;Y|ZW) = I(XW;Y|Z) = I(X;Y|Z) + I(W;Y|XZ).$$

The terms $I(W;Y|Z)$ and $I(W;Y|XZ)$ are 0 since $XY - Z - W$. ◀

The next proposition says that for any discrete random variable X there is a prefix-free encoding with average length at most $H(X) + 1$.

▶ **Proposition 3.4** (Huffman Encoding). *Let X and Y be random variables where X is discrete. Then, there exists a prefix-free encoding $\ell : \mathsf{supp}(X) \to \{0,1\}^*$ satisfying $\mathbb{E}_{xy}[|\ell(x)| \mid Y = y] \leq H(X|Y) + 1$.*

3.2 Common Information and Error-free Sampling

Wyner [35] defined the quantity *common information* between X and M as

$$\mathbb{C}(X;M) = \inf_{X-W-M} I(XM;W),$$

where the infimum is taken over all jointly distributed W such that, $X - W - M$ and W is supported over a finite set. Wyner showed that the above infimum is always achieved. By the data-processing inequality applied to the Markov chain $X - W - M$ it is easily seen that $\mathbb{C}(X;M) \geq I(X;M)$.

It turns out that the gap between $\mathbb{C}(X;M)$ and $I(X;M)$ can be very large. There are known examples of random variables X and M where $\mathbb{C}(X;M) = \omega(I(X;M))$. We include one simple example in Appendix A. Another example is described in the work of Harsha et al. [24], who also proved a related upper bound. They showed that there always exist C and S, where S is independent of X, $X - CS \to M$ and $H(C) \approx I(X;M)$. The random variable S in their work depends on the distribution of M. Braverman and Garg [8] showed a similar result that we quote and use in this work:

▶ **Lemma 3.5** ([8]). *Let $p(xm)$ be an arbitrary discrete probability distribution, with finite support. Let S be an infinite list of uniform samples from $\mathsf{supp}(M) \times [0,1]$, independent of XM. Then there exists a random variable C such that $X - CS \to M$ and $H(C|S) \leq I(X;M) + \log(I(X;M)+1) + \mathcal{O}(1)$.*

3.3 Streaming Algorithms

Without loss of generality, we associate the values stored by the algorithm with a non-negative integer. Assuming that the inputs to the algorithm come from the domain $\mathcal{X}$, a streaming algorithm defines a function $\mathcal{A} : [n] \times \mathbb{N} \times \mathcal{X} \to \mathbb{N}$. At time $t-1$, let the memory state of the algorithm be m_{t-1} (we define $m_0 := 1$). On seeing the input x_t at time t, the algorithm computes the t^{th} memory state $m_t := \mathcal{A}(t, m_{t-1}, x_t)$. The output of the algorithm is m_n. Randomized streaming algorithms toss *independent* random coins r_t at each time-step t and sample the memory state at time t as follows: $m_t := \mathcal{A}(t, m_{t-1}, r_t, x_t)$.

The following is obvious from the definition:

▶ **Proposition 3.6** (Markov Chain Property). *If $m_1, \ldots, m_n$ denote the memory of a (possibly randomized) streaming algorithm, then for each $t \in [n]$, $X_{\leq n} M_{<t} - X_t M_{t-1} - M_t$.*

The last proposition also implies the following.

▶ **Proposition 3.7.** *For a randomized streaming algorithm, the following holds,*

$$I(M_{\leq n}; X_{\leq n}) = I(M_1; X_1) + I(M_2; X_2|M_1) + \cdots + I(M_n; X_n|M_{n-1}).$$

Proof. Applying the chain-rule, we get

$$I(M_{\leq n}; X_{\leq n}) = \sum_{t=1}^{n} I(M_t; X_{\leq n}|M_{<t}) \leq \sum_{t=1}^{n} I(M_t; X_t X_{\leq n} M_{<t-1}|M_{t-1}).$$

The second inequality follows since $I(M_t; X_t X_{\leq n} M_{<t-1}|M_{t-1}) = I(M_t; M_{<t-1}|M_{t-1}) + I(M_t; X_{\leq n}|M_{<t}) + I(M_t; X_t|M_{<t} X_{\leq n})$ and mutual information is a non-negative quantity.

Applying the chain rule one more time, we have

$$\begin{aligned} I(M_{\leq n}; X_{\leq n}) &\leq \sum_{t=1}^{n} I(M_t; X_t X_{\leq n} M_{<t-1}|M_{t-1}) \\ &= \sum_{t=1}^{n} I(M_t; X_t|M_{t-1}) + \sum_{t=1}^{n} I(M_t; X_{\leq n} M_{<t-1}|X_t M_{t-1}). \end{aligned}$$

Proposition 3.6 implies that $X_{\leq n} M_{<t} - X_t M_{t-1} - M_t$ for every $t \in [n]$ and hence the second term on the right hand side is zero. ◀

The following proposition states that both the transitional and cumulative information content are upper bounded by the average memory.

▶ **Proposition 3.8.** *For a randomized streaming algorithm $\mathcal{A}$ with average memory M,*

$$\max\{\mathsf{IC}^{tr}(\mathcal{A}), \mathsf{IC}^{cum}(\mathcal{A})\} \leq M\,.$$

▶ **Definition 3.9** (Simulation)**.** We say that a streaming algorithm $\mathcal{A}_1$ simulates another algorithm $\mathcal{A}_2$ if for every input $x_1, \ldots, x_n$, the distribution on outputs is exactly the same in both algorithms.

In general it even makes sense to allow errors during simulation. Our simulations have no error, so we define simulation using the strong definition given above.

4 Compression and Direct Sums for Streaming Computation

The following is a natural strategy to prove our direct-sum theorem: given an algorithm that computes f^k correctly with probability $2/3$ on all the streams and uses average memory M, first show that there is some stream "with respect to" which the information content is M/k. Then derive a randomized streaming algorithm that computes f and has information content at most M/k as follows: embed the input stream at the location j about which the memory has small information and simulate the behavior of the algorithm on this stream by generating the other streams randomly, or to say alternately, sample from the distribution $p(m_n|X^{(j)} = x)$. The resulting algorithm would have information content at most M/k but would still use M bits of average memory. The last step would then be to give a way to simulate a streaming algorithm that has information content $\mathtt{I}$ with a streaming algorithm that uses average memory approximately $\mathtt{I}$.

For product distributions, we can show that if there exists an algorithm for computing k copies of f with memory M, then there is a randomized algorithm for computing a single copy of f with transitional and cumulative information content at most M/k. To prove our direct-sum result, we are able to show that algorithms with transitional information content I can be simulated with $\mathcal{O}(n\mathtt{I} + n)$ average memory which as discussed before is best possible. To give an optimal direct-sum result, one could still hope that streaming algorithms with cumulative information content I can be simulated with $\mathcal{O}(I)$ average memory. We discuss more about this possibility in Section 5.

4.1 Non-product Distributions and Correlated Randomness

Before we begin the proof of our compression and direct-sum results, we briefly discuss the difficulty that arises in dealing with non-product distributions. For proving a direct-sum result for non-product distributions using the above strategy, the natural way of using an algorithm that computes k copies of f to compute a single copy of f, is to embed our input stream at position j and generate other streams as randomness so that we can run the algorithm for k copies. The algorithm we get for computing f in this way uses randomness that is correlated across various time-steps if the input stream distribution is non-product.

Transitional information content is not a useful information measure for compressing such algorithms as the following example shows. We give an example of a function which require $\Omega(1)$ average memory, but can be computed by an algorithm that uses correlated randomness and has transitional information content $1/n$. Let $f(x) = \sum_{t=1}^{n} x_t \mod 2$. Consider the following algorithm that takes as input a random input stream x (each update x_t is a bit) and computes $f(x)$. The algorithm at time t uses randomness r_t where $r_1, \dots, r_t$ are correlated so that they satisfy $\sum_{t=1}^{n} r_t = 0 \mod 2$. At time t, the algorithm stores in its memory $\sum_{i=1}^{t}(x_t + r_t) \mod 2$ and at time $t = n$ outputs the last value stored in memory. Since $\sum_{t=1}^{n} r_t = 0 \mod 2$, the algorithm outputs $f(x)$. This algorithm has transitional information content $1/n$, but one can not hope to compute the parity of an n bit string without using $\Omega(1)$ bits of average memory.

4.2 Compressing Streaming Algorithms

In this section we show how algorithms with small transitional information content can be simulated with small average memory.

▶ **Theorem 4.1** (Restated). *Let $\mathcal{A}$ be a randomized streaming algorithm with $\mathsf{IC}^{tr}(\mathcal{A}) = \mathtt{I}$. Then there exists a randomized streaming algorithm $\mathcal{A}_{tr}$ with average memory $\mathcal{O}(n\mathtt{I} + n)$ that simulates $\mathcal{A}$.*

Let $m_1, \dots, m_n$ denote the memory states of the algorithm $\mathcal{A}$. Recall that Lemma 3.6 implies that for each $t \in [n]$, $X_{\le n}M_{<t} - X_tM_{t-1} - M_t$. Hence, to prove Theorem 4.1, it suffices to sample from $p(m_t|x_t, m_{t-1})$ if m_{t-1} has been sampled correctly. The compression algorithm will toss random coins to sample an infinite list s_t of samples from $\mathsf{supp}(M_t) \times [0,1]$ and then sample C_t (whose existence is guaranteed by Lemma 3.5) satisfying

$$X_t - C_tS_t|M_{t-1} \to M_t|M_{t-1}, \tag{4.1}$$

$$H(C_t|S_tM_{t-1}) = I(M_t;X_t|M_{t-1}) + \log(I(M_t;X_t|M_{t-1}) + 1) + \mathcal{O}(1). \tag{4.2}$$

The value of m_t determined by the sample c_t is distributed according to the distribution $\mathsf{p}(m_t|x_t, m_{t-1})$.

The algorithm will store the Huffman encoding (Proposition 3.4) of C_t conditioned on S_t and M_{t-1}. This encoding determines C_t given S_t and M_{t-1}, both of which are known to the algorithm at this time.

```
Randomized Streaming Algorithm $\mathcal{A}_{tr}$
  Input       : Stream $x \sim p(x)$
  Randomness  : $s_1, \dots, s_n$ where $s_i$ is an infinite sequence of uniform samples from $\mathsf{supp}(M_i) \times [0,1]$.
  // At time $t$: the content of the memory are some encodings of $c_{<t}$, where $c_i$ determines $m_i$ given $s_i$
     and $m_{i-1}$.
  1. Let $m_{t-1}$ be determined by $c_{t-1}$ and $s_{t-1}$. On input $x_t$, sample $c_t$ from the Markov chain in (4.1);
  2. Append the Huffman encoding of $c_t$ conditioned on $s_t$ and $m_{t-1}$ to the previous memory contents;
```

Note that the algorithm needs to store the encodings of all the previous $c_{\le t}$ at time t since in order to determine m_t uniquely, the value of m_{t-1} needs to be known which depends on the previous memory contents.

The following proposition is straightforward from (4.1).

▶ **Proposition 4.2.** *The algorithm $\mathcal{A}_{tr}$ simulates $\mathcal{A}$.*

Next we finish the proof of Theorem 4.1 by bounding the total memory used by $\mathcal{A}_{tr}$.

▶ **Lemma 4.3.** *The average memory used by $\mathcal{A}_{tr}$ is $\mathcal{O}(n\mathtt{I} + n)$.*

Proof. At time t, the expected number of bits appended to the memory (where the expectation is over the choice of $x_{\le t}$ and $s_{\le t}$) is bounded by $H(C_t|S_tM_{t-1})$. From (4.2), this is at most $2I(M_t; X_t|M_{t-1}) + \mathcal{O}(1)$. Hence, the number of bits stored in the memory at a time $t \in [n]$ is at most

$$\sum_{i=1}^{t}(2I(M_i; X_i|M_{i-1}) + \mathcal{O}(1)) \leq \sum_{i=1}^{n}(2I(M_i; X_i|M_{i-1}) + \mathcal{O}(1)) = 2n\mathtt{I} + \mathcal{O}(n).$$

Since this is true for every time-step t, the average memory is also upper bounded by $2n\mathtt{I} + \mathcal{O}(n)$. ◀

4.3 Direct Sum for Product Distributions

Recall that we want to prove the following theorem.

▶ **Theorem 4.4** (Direct Sum – Restated). *If p is product input distribution, then*

$$M(f^k) = \Omega\left(k\left(\frac{M(f)}{n} - 1\right)\right).$$

To prove the above we first show that if there is a deterministic algorithm for computing k copies of f with average memory M and error probability 1/3, then there is a randomized algorithm which computes a single copy of f with error at most 1/3 and has transitional information content at most M/k. Then, we apply Theorem 4.1 to compress this algorithm and get a contradiction if M is smaller than the right hand side in Theorem 4.4.

4.3.1 Computing f with Small Information

Let $\mathcal{A}$ be a *deterministic* streaming algorithm that uses average memory M and computes f^k on inputs sampled from p^k with error at most 1/3. Let $m_1, \ldots, m_n$ denote the memory states of the algorithm $\mathcal{A}$. Consider the following *randomized* algorithm $\mathcal{A}_{ran}$ that computes f with error at most 1/3 on inputs sampled from p. The algorithm chooses a random $j \in [k]$, embeds the input stream at position j and at time t, samples and stores the memory state m_t from the distribution $\mathsf{p}(m_t|x_t^{(j)} = x_t, m_{t-1})$.

```
Randomized Streaming Algorithm $\mathcal{A}_{ran}$

Input      : Stream $x$ sampled from $p(x)$
Randomness : $j$ uniformly drawn from $[k]$, streams $x^{(-j)}$
Output     : $f(x)$ with error at most $1/3$

1. Set Stream $x^{(j)}$ to be $x$;
2. At time $t$, use randomness $x_t^{(-j)}$ to sample $m_t$ from $\mathsf{p}(m_t|x_t^{(j)} = x_t, m_{t-1})$;
3. Output the answer of the algorithm on stream $j$;
```

Note that for any fixed value of j, the algorithm $\mathcal{A}_{ran}$ uses independent randomness $x_t^{(-j)}$ in each step as the input distribution p is product. We show that on average over the choice of j, the transitional and cumulative information content of the above algorithm is at most M/k.

▶ **Lemma 4.5.** $\mathbb{E}_j[\mathsf{IC}^{tr}(\mathcal{A}_{ran}|J = j)] \leq M/k$ *and* $\mathbb{E}_j[\mathsf{IC}^{cum}(\mathcal{A}_{ran}|J = j)] \leq M/k$.

Proof of Lemma 4.5. Conditioned on any event $J = j$, the transitional information content of $\mathcal{A}_{ran}$ is given by

$$\begin{aligned}
\mathsf{IC}^{tr}(\mathcal{A}_{ran}|J=j) &= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t \mid M_{t-1}, J=j) \\
&= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t^{(j)} \mid M_{t-1}, J=j) \quad \text{(with probability 1, } X^{(j)} = X) \\
&= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t^{(j)} \mid M_{t-1}) \qquad (M_t \text{ ind. of event } J=j).
\end{aligned}$$

Since the input stream comes from a product distribution, $X_t^{(1)}, \ldots, X_t^{(k)}$ are all independent conditioned on M_{t-1}. By Lemma 3.1, the term $I(M_t; X_t^{(j)} \mid M_{t-1})$ in the above sum is bounded by $I(M_t; X_t^{(j)} \mid X_t^{(<j)} M_{t-1})$. Taking an expectation over j, we get

$$\begin{aligned}
\mathbb{E}_j[\mathsf{IC}^{tr}(\mathcal{A}_{ran}|J=j)] &\le \mathbb{E}_j\left(\frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t^{(j)} \mid X_t^{(<j)} M_{t-1})\right) \\
&= \frac{1}{k}\left(\frac{1}{n}\sum_{t=1}^{n}\sum_{j=1}^{k} I(M_t; X_t^{(j)} \mid X_t^{(<j)} M_{t-1})\right)
\end{aligned}$$

From the chain rule the right hand side above equals

$$\frac{1}{k}\left(\frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t^{(1)} \ldots X_t^{(k)} | M_{t-1})\right) = \frac{1}{k}\mathsf{IC}^{tr}(\mathcal{A}) \le \frac{M}{k},$$

where the last inequality follows since the transitional information content is bounded by the average memory (Proposition 3.8).

Analogously, the cumulative information content of $\mathcal{A}_{ran}$ is given by

$$\begin{aligned}
\mathsf{IC}^{cum}(\mathcal{A}_{ran}|J=j) &= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_{\le t} \mid J=j) \\
&= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_{\le t}^{(j)} \mid J=j) \qquad \text{(with probability 1, } X^{(j)} = X) \\
&= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_{\le t}^{(j)}) \qquad (M_t \text{ind. of event } J=j).
\end{aligned}$$

Since $X^{(1)}, \ldots, X^{(k)}$ are all independent, by Lemma 3.1, the term $I(M_t; X_{\le t}^{(j)})$ is at most $I(M_t; X_{\le t}^{(j)} \mid X_{\le t}^{(<j)})$. Taking an expectation over j and using the chain rule, we get

$$\begin{aligned}
\mathbb{E}_j[\mathsf{IC}^{cum}(\mathcal{A}_{ran}|J=j)] &\le \frac{1}{k}\left(\frac{1}{n}\sum_{t=1}^{n}\sum_{j=1}^{k} I(M_t; X_{\le t}^{(j)} \mid X_{\le t}^{(<j)})\right) \\
&= \frac{1}{k}\left(\frac{1}{n}\sum_{t=1}^{n} I(M_t; X_{\le t}^{(1)} \ldots X_{\le t}^{(k)})\right) = \frac{1}{k}\mathsf{IC}^{cum}(\mathcal{A}) \le \frac{M}{k}. \qquad \blacktriangleleft
\end{aligned}$$

4.3.2 Direct-sum Theorem

With the above, we can now apply Theorem 4.1 to get Theorem 4.4.

Proof of Theorem 4.4. Let $\mathcal{A}$ be a streaming algorithm that computes f^k with error at most 1/3 and average memory M. By Lemma 4.5, there is an algorithm $\mathcal{A}_{ran}$ that uses randomness j and r, computes f with error at most 1/3 and satisfies $\mathbb{E}_j[\mathsf{IC}^{tr}(\mathcal{A}_{ran})|j] \leq M/k$. Applying Theorem 4.1 to $\mathcal{A}_{ran}$ gives us a randomized algorithm that uses random coins j and r' and computes f using average memory $\mathbb{E}_{j,r'}[\frac{1}{n}\sum_{t=1}^{n}|M_t|] = \mathcal{O}(nM/k+n)$.

Since the random coins j and r' are independent of the input, we can fix them to get a deterministic streaming algorithm with average memory $\mathcal{O}(nM/k+n)$. Since this must be at least $M(f)$, we have

$$\mathcal{O}\left(\frac{nM}{k}+n\right) \geq M(f).$$

Rearranging the above gives us that M is lower bounded by $\Omega\left(k\left(\frac{M(f)}{n}-1\right)\right)$. ◀

5 Towards Optimal Direct Sums

The algorithm $\mathcal{A}_{ran}$ that we gave in the last section also had cumulative information content at most M/k as shown in Lemma 4.5. Analogous to Theorem 4.4, the following result follows. We omit the proof since it is very similar to that of Theorem 4.4.

▶ **Theorem 5.1** (Restated). *If every algorithm with cumulative information* $\mathtt{I}$ *can be simulated using average memory* $\mathcal{O}(\mathtt{I})$, *then* $M(f^k) = \Omega(k\cdot(M(f)-1))$.

In this section, we describe a compression algorithm that could possibly simulate an algorithm with cumulative information content $\mathtt{I}$ with average memory $\mathcal{O}(\mathtt{I}+1)$. However, we are unable to either prove or disprove it.

To give some intuition about the new algorithm, let us recall Algorithm $\mathcal{A}_{tr}$ where the compression algorithm stored Huffman encodings (Proposition 3.4) of C_t satisfying $X_t - C_tS_t|M_{t-1} \to M_t|M_{t-1}$. This necessitated storing the whole history since to determine the sample m_t required knowing encodings of all the previous $c_{<t}$.

The new algorithm that we call $\mathcal{A}_{cum}$, on receiving the input x_t at time t, samples C_t conditioned on the value of x_t and m_{t-1} where C_t satisfies the following properties that follow from Lemma 3.5:

$$X_t - C_tS_t|S_{<t} \to M_t|S_{<t}, \tag{5.1}$$
$$H(C_t|S_{\leq t}) \leq I(M_t;X_tM_{t-1}|S_{<t}) + \log(I(M_t;X_tM_{t-1}|S_{<t})+1) + \mathcal{O}(1). \tag{5.2}$$

Again the value of m_t determined by the sample c_t is distributed according to the distribution $\mathsf{p}(m_t|x_t,m_{t-1})$. Moreover, the algorithm $\mathcal{A}_{cum}$ will store the Huffman encoding of C_t conditioned on $S_{\leq t}$ which avoids the need to store all the previous memory contents since $S_{\leq t}$ is randomness independent of the input and can be fixed in the beginning.

```
Randomized Streaming Algorithm $\mathcal{A}_{cum}$
  Input      : Stream $x \sim p(x)$
  Randomness : $s_1, \dots, s_n$ where $s_i$ is an infinite sequence of uniform samples from $\mathsf{supp}(M_i) \times [0,1]$.
  // At time $t$: the content of the memory are some encodings of $c_{<t}$, where $c_i$ determines $m_i$ given
     $s_{\leq i}$.
  1. Let $m_{t-1}$ be determined by $c_{t-1}$ and $s_{t-1}$. On input $x_t$, sample $c_t$ from the Markov chain in (5.1);
  2. Store the Huffman encoding of $c_t$ conditioned on $s_{\leq t}$;
```

▶ **Conjecture 5.2.** *Let $\mathcal{A}$ be a randomized streaming algorithm with $\mathsf{IC}^{cum}(\mathcal{A}) = \mathtt{I}$. Then, $\mathcal{A}_{cum}$ simulates $\mathcal{A}$ using $\mathcal{O}(\mathtt{I}+1)$ average memory.*

The proof that the above compression algorithm gives a correct simulation is straightforward from (5.1). We are able to prove the following bounds on the memory used by the above algorithm.

▶ **Lemma 5.3.** *In expectation over the choice of $s_{\leq t}$ and $x_{\leq t}$, the memory used by algorithm $\mathcal{A}_{cum}$ at time t is at most $\mathcal{O}(I(M_t;X_{\leq t}|S_{<t})+1)$.*

Proof. The memory used by algorithm $\mathcal{A}_{cum}$ at time t is bounded by $H(C_t|S_{\leq t})$ which as given by (5.2) is at most $\mathcal{O}(I(M_t;X_tM_{t-1}|S_{<t})+1)$. Moreover, since $M_{t-1}|S_{<t}$ is determined given $C_{t-1}|S_{<t}$,

$$I(M_t;X_tM_{t-1}|S_{<t}) \leq I(M_t;X_tC_{t-1}|S_{<t}),$$

by the data processing inequality (Proposition 3.2).

Next, we will show that $I(M_t;X_tC_{t-1}|S_{<t})$ is upper bounded by $I(M_t;X_{\leq t}|S_{<t})$. To show this we bound $I(M_t;X_tC_{t-1}|S_{<t}) \leq I(M_t;X_{\leq t}C_{t-1}|S_{<t})$ which by chain rule is,

$$\begin{aligned}&= I(M_t;X_{<t}C_{t-1}|S_{<t}) + I(M_t;X_t|S_{<t}X_{<t}C_{t-1})\\ &= I(M_t;X_{<t}|S_{<t}) + I(M_t;C_{t-1}|S_{<t}X_{<t}) + I(M_t;X_t|S_{<t}X_{<t}C_{t-1}).\end{aligned}$$

Note that in the algorithm $\mathcal{A}_{cum}$, $X_{<t}$ and $S_{<t}$ completely determine C_{t-1}. Hence, the second term in the above expression is 0. Moreover, by the same fact $M_tX_t - S_{<t}X_{<t} \to C_{t-1}$ and hence by Proposition 3.3, the last term $I(M_t;X_t|S_{<t}X_{<t}C_{t-1}) = I(M_t;X_t|X_{<t}S_{<t})$.

The above discussion yields that

$$\begin{aligned}I(M_t;X_tC_{t-1}|S_{<t}) &\leq I(M_t;X_{<t}|S_{<t}) + I(M_t;X_t|X_{<t}S_{<t})\\ &= I(M_t;X_{\leq t}|S_{<t}),\end{aligned}$$

where the second equality follows by another application of the chain rule. ◀

Note that since $S_{<t}$ is independent of $X_{\leq t}$, the above quantity $I(M_t;X_{\leq t}|S_{<t})$ is at least as large as $I(M_t;X_{\leq t})$ (recall Lemma 3.1), but it is possible that similar to Lemma 3.5, they could be the same up to some lower order error terms. Towards proving such a statement, we first propose to investigate whether the following stronger version of Lemma 3.5 holds.

▶ **Conjecture 5.4.** *Let X and M be arbitrary discrete random variables with finite support. Let S be an infinite list of samples from $\mathsf{supp}(M) \times [0,1]$. Then, there exist a random variable C such that*

- *$X - CS \to M$.*
- *$H(C|S) \leq I(M;X) + \log(I(M;X)+1) + \mathcal{O}(1)$.*
- *For any discrete random variable N such that $X - M - N$, it holds that*

$$I(N;M|S) \leq I(N;X) + \log(I(N;X)+1) + \mathcal{O}(1).$$

We also point out that an inductive use of the above conjecture does not give a non-trivial upper bound on the memory used by the algorithm $\mathcal{A}_{cum}$ because of the error terms in the last statement of the conjecture. But we hope that the techniques used in proving the above conjecture would be helpful in analyzing the memory used by the algorithm $\mathcal{A}_{cum}$. Nonetheless the above conjecture might be interesting in its own right and of potential use somewhere else.

Acknowledgments. We thank Paul Beame for helpful comments.

References

1 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

2 Chrisil Arackaparambil, Joshua Brody, and Amit Chakrabarti. Functional monitoring without monotonicity. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, volume 5555 of *Lecture Notes in Computer Science*, pages 95–106. Springer Berlin Heidelberg, 2009.

3 Brian Babcock and Chris Olston. Distributed top-k monitoring. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD'03, pages 28–39, New York, NY, USA, 2003. ACM.

4 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An Information Statistics Approach to Data Stream and Communication Complexity. In *FOCS*, pages 209–218, 2002.

5 Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. *SIAM J. Comput.*, 42(3):1327–1363, 2013.

6 Mark Braverman. Interactive information complexity. In *STOC*, pages 505–524, 2012.

7 Mark Braverman, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. A tight bound for set disjointness in the message-passing model. In *FOCS*, pages 668–677, 2013.

8 Mark Braverman and Ankit Garg. Public vs Private Coin in Bounded-Round Information. In *ICALP*, pages 502–513, 2014.

9 Mark Braverman and Anup Rao. Information equals amortized communication. In *FOCS*, pages 748–757, 2011.

10 Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*, pages 107–117, 2003.

11 Arkadev Chattopadhyay, Jaikumar Radhakrishnan, and Atri Rudra. Topology matters in communication. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 631–640, 2014.

12 G. Cormode, S. Muthukrishnan, and Wei Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 57–57, April 2006.

13 Graham Cormode. Sketching streams through the net: Distributed approximate query tracking. In *VLDB*, pages 13–24, 2005.

14 Graham Cormode and Minos Garofalakis. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *SIGMOD*, pages 25–36, 2005.

15 Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. *ACM Trans. Algorithms*, 7(2):21:1–21:20, March 2011.

16 Graham Cormode, S. Muthukrishnan, Ke Yi, and Qin Zhang. Optimal sampling from distributed streams. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS'10, pages 77–86, New York, NY, USA, 2010. ACM.

17 Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

18 Pavol Duris and Jose D.P. Rolim. Lower bounds on the multiparty communication complexity. *Journal of Computer and System Sciences*, 56(1):90–95, 1998.

19 Funda Ergun and Hossein Jowhari. On distance to monotonicity and longest increasing subsequence of a data stream. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'08, pages 730–736, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

20 Anna Gál and Parikshit Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. *SIAM J. Comput.*, 39(8):3463–3479, August 2010.

21 Anat Ganor, Gillat Kol, and Ran Raz. Exponential Separation of Information and Communication for Boolean Functions. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:113, 2014.

22 André Gronemeier. Asymptotically optimal lower bounds on the nih-multi-party information complexity of the and-function and disjointness. In *STACS 2009*, pages 505–516, 2009.

23 Sudipto Guha and Zhiyi Huang. Revisiting the direct sum theorem and space lower bounds in random order streams. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, volume 5555 of *Lecture Notes in Computer Science*, pages 513–524. Springer Berlin Heidelberg, 2009.

24 Prahladh Harsha, Rahul Jain, David A. McAllester, and Jaikumar Radhakrishnan. The communication complexity of correlation. *IEEE Transactions on Information Theory*, 56(1):438–449, 2010. doi:10.1109/TIT.2009.2034824.

25 Zengfeng Huang, Božidar Radunović, Milan Vojnović, and Qin Zhang. Communication complexity of approximate maximum matching in distributed graph data. In *STACS*, 2015.

26 Ram Keralapura, Graham Cormode, and Jeyashankher Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD'06, pages 289–300, New York, NY, USA, 2006. ACM.

27 Amit Manjhi, Vladislav Shkapenyuk, Kedar Dhamdhere, and Christopher Olston. Finding (recently) frequent items in distributed data streams. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE'05, pages 767–778, Washington, DC, USA, 2005. IEEE Computer Society.

28 Marco Molinaro, David P. Woodruff, and Grigory Yaroslavtsev. Beating the direct sum theorem in communication complexity with implications for sketching. In *SODA*, pages 1738–1756, 2013.

29 Jeff M. Phillips, Elad Verbin, and Qin Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'12, pages 486–501. SIAM, 2012.

30 Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Transactions on Database Systems*, 32(4), November 2007.

31 Izchak Sharfman, Assaf Schuster, and Daniel Keren. Shape sensitive geometric monitoring. In *Proceedings of the Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS'08, pages 301–310, New York, NY, USA, 2008. ACM.

32 David Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'04, pages 167–175, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

33 David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *STOC*, pages 941–960, 2012.

34 David P. Woodruff and Qin Zhang. An optimal lower bound for distinct elements in the message passing model. In *SODA*, pages 718–733, 2014.
35 A.D. Wyner. The common information of two dependent random variables. *IEEE Transactions on Information Theory*, 21(2):163–179, 1975.

A Separation between common information and mutual information

In this section we will give an explicit example of random variables X and M such that $\mathbb{C}(X;M) = \omega(I(X;M))$. Let G be a bipartite graph on the vertex set $([n],[n])$ such that the edge density of G is $\frac{1}{2} + o(1)$ and there are no cliques with more than $3n \log n$ edges in G. As the following lemma shows a random bipartite graph where each edge is picked with probability $1/2$ satisfies these properties with high probability, so such graphs exist.

▶ **Lemma 1.1.** *With probability $1 - o(1)$, a random bipartite graph on $([n],[n])$ where each edge is included with probability $1/2$ has no clique $U \times V$ where $U, V \subseteq [n]$ satisfying* $\min\{|U|,|V|\} \geq 2\log n + 2$.

Proof. Set $t := 2\log n + 2$ for notational convenience. If there is a clique $U \times V$ with $\min\{|U|,|V|\} \geq t$ then there also exists a clique of size $t \times t$. Consequently, to prove the lemma it suffices to upper bound the probability that a $t \times t$ clique exists in the graph. This probability is at most

$$\binom{n}{t}\binom{n}{t}2^{-t^2} \leq n^{2t}2^{-t^2} = 2^{2t\log n - t^2} = 2^{t(2\log n - t)} \leq 2^{-2t} = o(1). \qquad ◀$$

A corollary of the above lemma is that the maximal clique in a random bipartite graph with edge probability $1/2$ has at most $n \cdot 3\log n$ edges with high probability. Also it is easy to see that with probability $1 - o(1)$, every vertex in a random bipartite graph with edge probability $1/2$ has degree between $\frac{n}{2} - o(n)$ and $\frac{n}{2} + o(n)$.

Now we can describe the random variables X and M which will be the end points of a uniformly random edge E in the graph G. It is easily seen that the mutual information $I(X;M) \leq 1 - o(1)$ since $H(X) = \log n$ while for any $M = m$, $H(X|M = m) \geq \log n - 1 - o(1)$. On the other hand, if $X - W - M$, then for any value w attained by W, $\mathsf{supp}(X|W = w)$ and $\mathsf{supp}(M|W = w)$ has to form a clique in the graph G. Since the maximal clique in G has at most $3n\log n$ edges, for any $W = w$, it holds that

$$|\mathsf{supp}(X|W = w)| \cdot |\mathsf{supp}(M|W = w)| \leq 3n\log n.$$

It follows that for any such W we can write

$$H(XM|W) \leq \log(|\mathsf{supp}(X|W = w)| \cdot |\mathsf{supp}(M|W = w)|) = \log n + \mathcal{O}(\log\log n).$$

Hence we have that the mutual information between XM and W is,

$$\begin{aligned} I(XM;W) = H(XM) - H(XM|W) &\geq (2\log n - 1 - o(1)) - (\log n + \mathcal{O}(\log\log n)) \\ &= \log n - \mathcal{O}(\log\log n), \end{aligned}$$

for any W satisfying $X - W - M$. It follows that $\mathbb{C}(X;M) = \Omega(\log n)$ while $I(X;M) \leq 1 - o(1)$.

Explicit List-Decodable Codes with Optimal Rate for Computationally Bounded Channels

Ronen Shaltiel[*1] **and Jad Silbak**[†2]

1 **Department of Computer Science, University of Haifa, Israel**
`ronen@cs.haifa.ac.il`

2 **Department of Computer Science, University of Haifa, Israel**
`jadsilbak@gmail.com`

Abstract

A stochastic code is a pair of encoding and decoding procedures (Enc, Dec) where $\text{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$. The code is (p, L)-list-decodable against a class $\mathcal{C}$ of "channel functions" $C : \{0,1\}^n \to \{0,1\}^n$ if for every message $m \in \{0,1\}^k$ and every channel $C \in \mathcal{C}$ that induces at most pn errors, applying Dec on the "received word" $C(\text{Enc}(m, S))$ produces a list of at most L messages that contain m with high probability over the choice of uniform $S \leftarrow \{0,1\}^d$. Note that both the channel C and the decoding algorithm Dec *do not* receive the random variable S, when attempting to decode. The rate of a code is $R = k/n$, and a code is explicit if Enc, Dec run in time poly(n).

Guruswami and Smith (J. ACM, to appear), showed that for every constants $0 < p < \frac{1}{2}$ and $c > 1$ there are Monte-Carlo explicit constructions of stochastic codes with rate $R \geq 1 - H(p) - \epsilon$ that are $(p, L = \text{poly}(1/\epsilon))$-list decodable for size n^c channels. Monte-Carlo, means that the encoding and decoding need to share a public uniformly chosen poly(n^c) bit string Y, and the constructed stochastic code is (p, L)-list decodable with high probability over the choice of Y.

Guruswami and Smith pose an open problem to give fully explicit (that is not Monte-Carlo) explicit codes with the same parameters, under hardness assumptions. In this paper we resolve this open problem, using a minimal assumption: the existence of poly-time computable pseudorandom generators for small circuits, which follows from standard complexity assumptions by Impagliazzo and Wigderson (STOC 97).

Guruswami and Smith also asked to give a fully explicit unconditional constructions with the same parameters against $O(\log n)$-space online channels. (These are channels that have space $O(\log n)$ and are allowed to read the input codeword in one pass). We resolve this open problem.

Finally, we consider a tighter notion of explicitness, in which the running time of encoding and list-decoding algorithms does not increase, when increasing the complexity of the channel. We give explicit constructions (with rate approaching $1 - H(p)$ for every $p \leq p_0$ for some $p_0 > 0$) for channels that are circuits of size $2^{n^{\Omega(1/d)}}$ and depth d. Here, the running time of encoding and decoding is a fixed polynomial (that does not depend on d).

Our approach builds on the machinery developed by Guruswami and Smith, replacing some probabilistic arguments with explicit constructions. We also present a simplified and general approach that makes the reductions in the proof more efficient, so that we can handle weak classes of channels.

1998 ACM Subject Classification F.1.3. Complexity Measures and Classes

Keywords and phrases Error Correcting Codes, List Decoding, Pseudorandomness

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.45

[*] Research supported by ERC grant 279559, BSF grant 2010120, and ISF grant 864/11.
[†] Research supported by ERC grant 279559.

1 Introduction

List decodable codes

List decodable codes are extensively studied in Coding Theory and Theory of Computer Science, and have many applications. In the paragraph below we define list-decodable codes, using a functional view, which is more convenient for this paper.

A code is defined by a pair $(\mathrm{Enc}, \mathrm{Dec}))$ of encoding and decoding procedures. We say that $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$, is (p, L)-list decodable, if there exits a function Dec which given $y \in \{0,1\}^n$, $\mathrm{Dec}(y)$ produces a list of size L containing all elements $m \in \{0,1\}^k$ such that $\delta(y, \mathrm{Enc}(m)) \le p$, (here $\delta(x, y)$ is the relative hamming distance of x and y). Unique decoding is the special case where $L = 1$, and a code is explicit if both encoding and decoding can be performed in time polynomial in n. The rate of a code is $R = \frac{k}{n}$. (A more detailed formal definition is given in Section 3.2).

Towards explicit capacity-achieving, binary list decodable codes

It is known that for $0 < p < \frac{1}{2}$, binary (p, L)-list decodable codes must have rate $R \le 1-H(p)$ for nontrivial size lists, and a longstanding open problem in coding theory is to give an explicit construction of binary codes matching list-decoding capacity. That is, show that for every constants $0 < p < \frac{1}{2}$, and $\epsilon > 0$, and for every sufficiently large n, there are explicit binary list decodable codes with rate $R = 1 - H(p) - \epsilon$, that are (p, L)-list decodable, for a constant L that depends on ϵ. The probabilistic method shows that there exist nonexplicit codes with these parameters. (In fact, the probabilistic method achieves list size L which is $\mathrm{poly}(1/\epsilon)$, and this is a benchmark that can be compared to.) Today, despite substantial effort, no explicit constructions are known, even if we insist only on explicit encoding, and do not require list-decoding to be explicit.

Restricted channels

Explicit uniquely decodable, binary codes achieving rate approaching $1 - H(p)$, are known for restricted classes of channels. There is a large body of work in Shannon's framework, on channels which are not *adversarial* and inflict "random errors". The most famous example is a binary symmetric channel, that flips each symbol independently with probability p, and there are explicit, uniquely decodable, binary codes with rate approaching $1 - H(p)$ for such channels.

Computationally bounded channels

Lipton [11] considered intermediate classes of *adversarial* channels according to the computational complexity of the channel. More specifically, we can think of a channel as a function $C : \{0,1\}^n \to \{0,1\}^n$ and consider families channels that:

- Induce at most pn errors. That is, for every $z \in \{0,1\}^n$, $E_C(z) := z \oplus C(z)$ has hamming weight at most pn.
- Are computationally bounded. That is, we only consider C that belong to some complexity class $\mathcal{C}$.

Natural examples of complexity classes are polynomial size circuits and logarithmic space branching programs. Note that these two classes are nonuniform, and it is more natural to use nonuniform classes, as such classes trivially contain channels C where E_C is constant (meaning that there is a fixed error vector e such that $C(z) = z \oplus e$). Such channels are called “additive channels” and as they are the simplest form of adversarial behavior, it makes sense that we allow them in any class of computationally bounded channels.

Another advantage of using nonuniform classes of channels, is that it is sufficient to consider *deterministic* channels, in order to obtain security against *randomized* channels. This is because by averaging, if there is a computationally bounded randomized channel that is able to prevent decoding on some message m, then we can fix its random coins and obtain a deterministic channel (which is hardwired with a good choice of random coins).

1.1 Stochastic codes

Unfortunately, the notion of computationally bounded channels is not interesting in the standard setup of error correcting codes: It is easy to show that if a code $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$ is list-decodable against additive channels, then it is list-decodable against unbounded channels.[1]

Several setup assumptions were introduced in order to circumvent this problem. In this paper, we are interested in a setup of “stochastic codes” studied by Guruswami and Smith [5]. We remark that other setups have been considered and we mention these in Section 1.4.

Let $\mathcal{C}$ be a class of channels that induce at most pn errors. A stochastic code against $\mathcal{C}$ consists of a pair of algorithm $(\mathrm{Enc}, \mathrm{Dec})$ such that:

- The encoding algorithm $\mathrm{Enc}(m, S)$ receives a message $m \in \{0,1\}^{Rn}$ and a uniform string S (that is not known to the channel or decoding algorithm) and outputs an n bit string that is the codeword.
- A channel $C \in \mathcal{C}$, that **does not** receive the string S, corrupts the codeword, generating $C(\mathrm{Enc}(m, S))$.
- The decoding algorithm gets the “corrupted codeword” $C(\mathrm{Enc}(m, S))$, but **does not** receive the string S.
- For every message m, and for every channel $C \in \mathcal{C}$, the decoding done by $\mathrm{Dec}(C(\mathrm{Enc}(m, S)))$ needs to successfully recover the original message m with probability $1 - \nu$ over the choice of S. ($\nu > 0$ is an error parameter).

  Here, "success" means to output m (in case of unique decoding) or output a list of size L that contains m (in case of list decoding).

A formal definition follows:

▶ **Definition 1** (Stochastic Codes)**.** Let k, n, q be parameters and let $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ be a function. Let $\mathcal{C}$ be a class of functions from n bits to n bits. We say that Enc is an encoding function for a stochastic code that is:

- **decodable** with success probability $1 - \nu$ against channels in $\mathcal{C}$, if there exists a function $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^k$ such that for every $m \in \{0,1\}^k$ and every $C \in \mathcal{C}$, $\Pr_{S \leftarrow U_d}[\mathrm{Dec}(C(\mathrm{Enc}(m, S))) = m] \geq 1 - \nu$. We typically, parameterize $\mathcal{C}$ with two parameters: the complexity of functions in the class, and the number errors that they induce.
- $\boldsymbol{L}$**-list-decodable** with success probability $\boldsymbol{1 - \nu}$ against channels in $\boldsymbol{\mathcal{C}}$ if the function Dec above is allowed to output a list of size at most $\boldsymbol{L}$ that contains $\boldsymbol{m}$.

[1] Specifically, if a code is not (combinatorially) list-decodable, then there exist a received word y that has too many codewords that are close to it. Let c be one of these codewords, and let $e = c \oplus y$ and consider the additive channel $C_e(z) = z \oplus e$. This channel “breaks” the code as $C(c) = c \oplus e = y$, and y is a received word on which decoding cannot succeed.

A code is **explicit** if its encoding and decoding functions are computable in time polynomial in their input and output. The rate of the code is the ratio of the message length and output length of Enc.

Guruswami and Smith [6] gave explicit constructions of stochastic codes with rate approaching $1 - H(p)$ (for $0 < p < \frac{1}{2}$) that are uniquely decodable against additive channels. They also showed that for $p > 1/4$ there are computationally weak channel families, against which, stochastic codes with rate approaching $1 - H(p)$ and unique decoding do not exist. (All the complexity classes considered in this paper can simulate these weak channels.)

A Monte-Carlo construction of stochastic codes for poly-size circuits

Guruswami and Smith [6] showed that for every constant c, there is a Monte-Carlo explicit construction of list decodable stochastic codes against channels of size n^c, with rate approaching $1 - H(p)$. By Monte-Carlo, we mean that:

- The encoding and decoding algorithms receive an additional input y of length poly(n^c).
- With high probability over the choice of y, the encoding and decoding algorithms (that are hardwired with y) form the required stochastic code.[2]

1.2 Our results

Guruswami and Smith stated the following open problem: give fully explicit (that is *not* Monte-Carlo) constructions of stochastic codes against poly-size circuits, under complexity theoretic assumptions.

Necessity of complexity theoretic assumptions

As we explain later, complexity theoretic assumptions are not necessary in order to give *Monte-Carlo* constructions of stochastic codes. They are necessary to give fully explicit constructions (which are not Monte-Carlo) in the following sense: Given a stochastic code against circuits of size n^c, we can consider the "optimal channel" that given a codeword $z \in \{0,1\}^n$, tries all possible error vectors $e \in \{0,1\}^n$ of relative hamming weight p, and finds the first one on which decoding fails, if such a vector exist. This channel succeeds iff the code isn't secure against unbounded channels. If the code isn't secure against unbounded channels (but secure against size n^c channels) then this attack cannot be carried out in size n^c. This means that there is a problem computable in $\mathrm{E} = DTIME(2^{O(n)})$ that for every sufficiently large n, cannot be solved by size n^c circuits.[3] We remark that this type of assumptions (namely, that there is a problem in E that requires large circuits) is exactly the type of assumption that implies and is implied by, existence of explicit pseudorandom generators in the Nisan-Wigderson setting [15, 8].

1.2.1 Explicit stochastic codes for poly-size circuits

Our first result resolves the open problem posed by Guruswami and Smith, and we construct explicit stochastic codes against poly-size circuit channels, under an assumption that is only slightly stronger than what is implied by the existence of such codes.

[2] We mention that the approach of Guruswami and Smith dictates that the length of y is larger than n^c (and in general larger than the log of the number of allowed channels). This means that a channel is not "sufficiently complex" to receive y as input.

[3] In fact, for this argument, we don't need the stochastic code to be explicit. Encoding is allowed to be arbitrary, and decoding is allowed to run in time $2^{O(n)}$.

▶ **Theorem 2** (Explicit stochastic codes for poly-size channels). *If there exists a constant $\beta > 0$ and a problem in $E = DTIME(2^{O(n)})$ such that for every sufficiently large n, solving E on inputs of length n, requires circuits of size $2^{\beta \cdot n}$, then for every constants $0 < p < \frac{1}{2}$, $\epsilon > 0$, $c > 1$, and for every sufficiently large n, there are explicit stochastic codes with rate $1 - H(p) - \epsilon$ that are L-list decodable for size n^c circuits that induce at most pn-errors, where $L = poly(1/\epsilon)$ is a constant.*

Theorem 2 is stated in more detailed form in Theorem 32. The assumption used in the Theorem is a standard complexity assumption, and was used by Impagliazzo and Wigderson [8] to show that BPP=P.

1.2.2 Unconditional explicit stochastic codes for space $O(\log n)$ online channels

Guruswami and Smith also considered "space s online channels". These are channels $C : \{0,1\}^n \to \{0,1\}^n$ implemented by space s (or equivalently width 2^s) oblivious read-once branching programs (ROBPs). Below is a standard definition of ROBPs tailored for functions that output many bits.

Read Once Branching Programs

We will only be interested in space $s \geq \log n$. A space s ROBP $C : \{0,1\}^n \to \{0,1\}^n$ is defined using a layered graph with $n+1$ layers, where the first layer has a single node v_0, and remaining layers have 2^s nodes. Each node v in the first n layers has two outgoing edges (labeled with zero and one) connected to nodes in the next layer, and each node v is also labeled by an "output bit" $b(v)$. On input $x \in \{0,1\}^n$, the computation of C is defined by following the unique path from v_0 to the last layer, defined by taking the edge marked with x_i at step i. The output $C(x)$ is the concatenation of the n output bits, collected at nodes along the path. It is standard that for $s \geq \Omega(\log n)$ ROBPs with space $O(s)$ capture the nonuniform version of space $O(s)$ computation, that reads its n bit input x in fixed order. We remark that all the results in this paper also hold if we allow channels to have s bits of "lookahead", allowing them to also read the bits $i+1, \ldots, i+s$ before outputting the i'th bit.

Guruswami and Smith stated the following open problem: give unconditional fully explicit (that is *not* Monte-Carlo) constructions of stochastic codes against space $O(\log n)$ online channels.[4] Our second result resolves this open problem.

▶ **Theorem 3** (Explicit stochastic codes for space $O(\log n)$ online channels). *For every constants $0 < p < \frac{1}{2}$, $\epsilon > 0$, $c > 1$, and for every sufficiently large n, there are explicit stochastic codes with rate $1 - H(p) - \epsilon$ that are L-list decodable for space $c \log n$ online channels that induce at most pn-errors, where $L = poly(1/\epsilon)$ is a constant.*

Theorem 3 is stated in more detailed form in Theorem 33.

[4] A preliminary version of [6] contained an unconditional Monte-Carlo construction of stochastic code against space $O(\log n)$ online channels, and a conditional Monte-Carlo construction for size n^c circuits (relying on the existence of "Nisan-Wigderson style", pseudorandom generators for size n^c circuits). However, Monte-Carlo constructions can easily obtain "Nisan-Wigderson style" pseudorandom generators, as a random function with polynomial size description is w.h.p. such a generator. Consequently, no hardness assumption is needed for Monte-Carlo constructions against polynomial size circuits, which are secure also against $O(\log n)$ space online channels.

Efficiency of encoding/decoding versus channel complexity

The approach of Guruswami and Smith [6] (that we also use) dictates that security can only be proven for channel families that are not sufficiently strong to run the decoding algorithm.[5] Consequently, in the Monte-Carlo construction and our Theorem 2, the running time of encoding and decoding is a polynomial in n that is larger than the circuit size n^c. It is an intriguing open problem whether stochastic codes with rate approaching $1 - H(p)$, that can be encoded and decoded in fixed polynomial time (say n^3) against any polynomial size channel, can be constructed (under cryptographic assumptions). We do not know whether this is possible.

We can however expect to obtain fixed polynomial time (that does not depend on the constant c) for encoding and decoding in our Theorem 3. Unfortunately, this is not the case, and the encoding and decoding algorithm that we obtain in Theorem 3 run in time polynomial in n^c (and in particular larger than n^c) when working against space $c \log n$ channels. We do not know how to avoid this dependence.

1.2.3 Stochastic codes for AC^0 channels, with fixed poly-time encoding/decoding

We are able to obtain fixed polynomial time algorithms for encoding and decoding for a family of channels implemented by superpolynomial size and constant depth circuits. For technical reasons, we achieve this only for $p \leq p_0$ for some $p_0 > 0$. The result is stated below.

▶ **Theorem 4** (Explicit stochastic codes for AC^0 channels). *There exist constants $p_0 > 0$ and $a > 0$ such that for every constants $0 < p \leq p_0$, $\epsilon > 0$, d, and for every sufficiently large n, there are explicit stochastic codes with rate $1 - H(p) - \epsilon$ that are L-list decodable for size $2^{n^{\frac{1}{ad}}}$ circuits of depth d that induce at most pn-errors, where $L = poly(1/\epsilon)$ is a constant. (Here, encoding and decoding run in fixed polynomial time that does not depend on d, and only the choice of which n is sufficiently large, depends on d.)*

The constant p_0 comes from a specific construction of AG-codes, and it seems that p_0 can be pushed to be any constant strictly smaller than $1/12$. Theorem 4 is stated in more detailed form in Theorem 34.

1.3 Perspective

Explicit codes against computationally bounded channels give the "best of both worlds": They can recover from errors induced by *adversarial* channels, while having information theoretic optimal rate approaching $1 - H(p)$.

As pointed out by Guruswami and Smith, essentially all randomized channels studied in the Shannon framework of error correcting codes, are computationally simple (and it seems that all of them can be implemented by constant depth circuits or online logspace). This means that the computational perspective leads to a unified construction of explicit codes that are good for all "Shannon style" randomized channels simultaneously, while also being able to recover against many adversarial channels (and in particular against additive channels).

[5] The approach of Guruswami and Smith (that we also use) relies on the fact that channels cannot distinguish between encodings of two messages. Therefore, if decoders aren't stronger than channels, they cannot hope to decode, even if there are no errors.

We believe that the distinction we make above (namely, whether encoding/decoding efficiency is allowed to increase with the complexity of the channel) is important so that the added benefit of codes for computationally bounded channels doesn't come with a price tag of being less efficient. Specifically, our construction for AC^0 channels uses "regular" coding theoretic ingredients and does not have to "pay extra" for being able to handle channels that are superpolynomial size circuits of constant depth.

An intriguing open problem is whether unique decoding is possible for computationally bounded channels with rate approaching $1 - H(p)$. Guruswami and Smith [6] showed that this is impossible for $p > 1/4$ (and their argument works for all classes of channels discussed in this paper). It is not known whether unique decoding is possible for $p < 1/4$ for the channel classes that we consider.

1.4 Some related work

The notion of computationally bounded channels was initially studied in cryptographic setups. We mention some of these works below.

Shared private randomness

We start with the notion of codes with "shared private randomness". While this setup was considered before the notion of stochastic codes, in this paper, it is natural view it as a version of stochastic codes in which the decoding algorithm **does** receive the S.

This corresponds to a standard symmetric cryptography setup in which honest parties (the encoder and decoder) share a uniform private key S, and the bad party (the channel) does not get the key.

Lipton [11] and following work (see [19] for more details) gave explicit constructions of uniquely decodable codes against computationally bounded channels, with rate approaching $1 - H(p)$, under cryptographic assumptions.

Note that the setup of stochastic codes is lighter. The encoder and decoder do not need to share a private random key. Moreover, a fresh key can be chosen on the spot every time the encoder encodes a message.

We also point out that the Monte-Carlo construction of Guruswami and Smith, also requires less setup. While the encoder and decoder do need to share a random string, this string does not need to be private. It can be chosen once and revealed to the channel.

Private Codes

A related notion of "private codes" was studied by Langberg [10]. Here channels are unbounded, codes are existential (and not explicit), and the focus is on minimizing the length of the shared key. Langberg provides asymptotically matching upper and lower bounds of $\Theta(\log n + \log(1/\nu))$, on the amount of randomness that needs to be shared for unique decoding in this setup.

Public key setup

Micali et al. [12] considered computationally bounded channels, and a cryptographic, public key setup. Their focus is to use this setup to convert a given (standard) explicit list-decodable code into an explicit uniquely decodable codes (in this specific public key setup).

2 Overview of the technique

In this section we give a high level overview of the construction. Our construction heavily relies on previous work in the area (mainly on that of Guruswami and Smith [6]). In this high level overview we attempt to highlight our technical contribution, while also giving a high level overview of the many ideas from previous work that are used in the construction. Therefore, we start with a high level description of earlier work, and build up to the work of Guruswami and Smith. Along the way, in Section 2.2 we explain the modifications that allow us to handle weak classes of channels. Finally, in Section 2.4, we present a self contained problem (that of constructing inner stochastic codes). Constructing such explicit codes is the main source of our improvement over Guruswami and Smith, and we give a high level overview of our approach.

The reader can skip this high level overview and go directly to the technical section.

2.1 Codes for the setup of shared private randomness

We start by explaining how to construct codes with rate approaching $1 - H(p)$ in the case that the setup allows shared private randomness. Recall that this can be thought of as a stochastic code in which the decoding algorithm receives the random string chosen by the encoding. We present the ideas that are used to construct codes against bounded channels in this setup, in two steps. We first explain how to handle additive channels, and then explain how this method can be extended to handle bounded channels that are not additive. The ideas from both these reductions are key components in the construction of Guruswami and Smith.

Reducing additive channels to binary symmetric channels

We start by constructing codes with shared private randomness against additive channels. The encoder and decoder will share a description S_π of a uniformly selected permutation $\pi : [n] \to [n]$. The encoding will be defined by

$$\mathrm{Enc}(m, S_\pi) = \pi(\mathrm{Enc}_{BSC}(m)),$$

meaning that Enc encodes m by a code for binary symmetric channels, and then uses the permutation π to rearrange the n indices of the encoding, placing the i'th bit, in the $\pi(i)$'th position. Note that for any additive channel $C_e(z) = z \oplus e$ that induces pn errors, the effect of the channel on $\mathrm{Enc}(m, S_\pi)$ can be essentially viewed as applying a binary symmetric channel on $\mathrm{Enc}_{BSC}(m)$, meaning that the decoder is able to uniquely decode against additive channels, with a code that has rate approaching $R = 1 - H(p)$ (which can be achieved explicitly for binary symmetric channels).

Smith [19] showed that an (almost) t-wise independent permutation can be coupled with specific constructions of codes for binary symmetric channels, and used instead of a truly random permutation. This reduces the length of the shared key and allows keys shorter than n.

Reducing computationally bounded channels to additive channels

It is possible to use cryptography (or more generally pseudorandomness) to handle computationally bounded channels: Assume that in addition to the seed S_π, the encoder and decoder also share a seed S_{PRG} for a pseudorandom generator PRG that fools computationally bounded channels and outputs n bits, and define:

$$\mathrm{Enc}'(m,(S_\pi,S_{PRG})) = \mathrm{Enc}(m,S_\pi) \oplus PRG(S_{PRG}) = \pi(\mathrm{Enc}_{BSC}(m)) \oplus PRG(S_{PRG}).$$

This means that the rate of Enc' is inherited from Enc and can approach $1-H(p)$. A useful property is that for every fixed s_π, the random variable $\mathrm{Enc}(m,(s_\pi,S_{PRG}))$ is pseudorandom for the channel. This can be used to show that a computationally bounded channel cannot prevent correct decoding.

We now explain this argument. The decoding algorithm $\mathrm{Dec}'(y,(s_\pi,s_{PRG}))$ will simply compute $y' = y \oplus PRG(s_{PRG})$ and apply the previous decoding algorithm Dec on y' and s_π. We show that for every computationally bounded channel C that induces at most pn errors, the decoding succeeds with probability at least $1-(\nu+\epsilon_{PRG})$, where ϵ_{PRG} is the error of the generator PRG.

We consider the function $A(m,s_\pi,e)$ that checks if $\mathrm{Dec}_{BSC}(\mathrm{Enc}(m,s_\pi)\oplus e)$ successfully recovers m. In the previous section we've seen that for every message m, and error vector e of relative hamming weight at most p, $\Pr[A(m,S_\pi,e)] \geq 1-\nu$. Consequently, for every channel C that induces pn errors,

$$\Pr[A(m,S_\pi,E_C(U_n))=1] \geq 1-\nu$$

(this follows as U_n is independent of S_π, and recall that $E_C(z) = z \oplus C(z)$). If decoding does not work, and there exist a message m such that:

$$\Pr[A(m,S_\pi,E_C(\mathrm{Enc}'(m,(S_\pi,S_{PRG}))))=1] < 1-(\nu+\epsilon_{PRG}).$$

By averaging over S_π, this gives that there exists a fixed value s_π such that:

$$\Pr[A(m,s_\pi,E_C(U_n))=1] - \Pr[A(m,s_\pi,E_C(\mathrm{Enc}'(m,(s_\pi,S_{PRG}))))=1] > \epsilon_{PRG},$$

meaning that $f(z) = A(m,s_\pi,E_C(z))$ distinguishes $\mathrm{Enc}(m,(s_\pi,S_{PRG}))$ from U_n with probability ϵ_{PRG}, which is a contradiction if PRG is ϵ_{PRG}-pseudorandom against f (which is essentially the composition of the channel and Dec_{BSC}). As Dec_{BSC} runs in polynomial time, it follows that a PRG against poly-size circuits suffices to handle poly-size channels.

2.2 A more efficient reduction for online logspace and AC^0

A drawback of the approach described above is that while the decoding algorithm Dec_{BSC} runs in polynomial time, existing constructions rely on decoding an "outer code" (typically, Reed-Solomon) which cannot be done by small constant depth circuits or small space ROBPs. In this paper we are interested in channels that run in online logspace or AC^0. We would like to use PRGs that fool these classes (and explicit constructions are unconditional) rather than PRGs for poly-size circuits (which are inherently conditional as they imply circuit lower bounds).

For this purpose we replace the code $(\mathrm{Enc}_{BSC},\mathrm{Dec}_{BSC})$ (for binary symmetric channels) by a code $(\mathrm{Enc}_{\mathrm{balanced}},\mathrm{Dec}_{\mathrm{balanced}})$ that is list decodable from *balanced errors*. We now define this notion. A string $e \in \{0,1\}^n$ is (b',p,γ)-balanced if when viewed as $e \in (\{0,1\}^{b'})^{n/b'}$, at most a γ fraction of blocks of e, have relative hamming weight larger than p. It is not hard to construct explicit codes which are list-decodable (with constant size lists) against error vectors that are (b',p,γ)-balanced and have rate approaching $1-H(p)$ for small constant γ. We give such a construction in Section 3.2.1.

If we take an error vector of Hamming weight p and permute it using a random (or t-wise independent) permutation, then with high probability it will indeed be $(b', p+\alpha, \gamma)$-balanced for sufficiently large b', and small constant $\alpha, \gamma > 0$. This means that codes against balanced errors in particular work against binary symmetric channels.

The advantage of this notion is that the function A of the previous section can be made more efficient. Rather than having to decode Enc_{BSC}, it is sufficient to check if the error vector e is $(b', p+\alpha, \gamma)$-balanced, which can be performed by models that can count (or even only approximately count) such as small ROBPs, or AC^0. This leads to more efficient reductions that enable us to use PRGs for weaker classes.[6]

2.3 Stochastic codes for bounded channels

We start by presenting the approach of Guruswami and Smith [6] to take codes for shared private randomness (as presented in the previous section) and convert them into stochastic codes.

Let $(\mathrm{Enc}', \mathrm{Dec}')$ be the code for shared private randomness (presented in the previous section). We will reserve N for the block length of the stochastic code that we want to construct, and use N_{data} as the block length of Enc'. We have that the rate of Enc' can approach $1-H(p)$ and so it is sufficient that the rate of the code $(\mathrm{Enc}, \mathrm{Dec})$ that we construct, approaches that of Enc'.

We will set $N = N_{\mathrm{data}} + N_{\mathrm{ctrl}}$ where $N_{\mathrm{ctrl}} = \epsilon \cdot N$ (for a small constant ϵ) so that the rate indeed approaches $1 - H(p)$. Loosely speaking, when a given a message m and "control information" S (which will include (S_π, S_{PRG}) as well as additional randomness) we will set $c_{\mathrm{data}} = \mathrm{Enc}'(m, (S_\pi, S_{PRG})) \in \{0,1\}^{N_{\mathrm{data}}}$ and $c_{\mathrm{ctrl}} \in \{0,1\}^{N_{\mathrm{ctrl}}}$ will be an encoding of S (that we specify later). We will then merge these two strings into a string $c = (c_{\mathrm{data}}, c_{\mathrm{ctrl}})$ of length N.

The high level intuition, is that the encoder encodes the control information S and embeds it in the encoding of m, hoping that the decoder can find it, decode it to get S, and then use the decoding algorithm Dec' (which requires S) to decode the data part.

However, there are two seemingly contradicting requirements: On the one hand, the decoder needs to find the "control information" in order to recover S. On the other, if it is easy to identify which part of the encoding encodes the "control information", then the channel can focus its errors on it, wiping it out completely.

Stochastic codes for additive channels

The first step taken by Guruswami and Smith is to ensure that an additive channel cannot wipe out the control information. For this purpose they divide the N output bits into $n = N/b$ blocks of length b (where b is a parameter to be chosen later). The encoder will use additional randomness S_{samp} to choose a random set $I = \{i_1, \ldots, \epsilon \cdot n\}$ of distinct indices in $[n]$. The string S_{samp} will be part of the "control information" S (making $S = (S_{\mathrm{samp}}, S_\pi, S_{PRG})$) and in order to make its length less than n, the sampling is made by a randomness efficient averaging sampler (see Section 3.3 for details). We will pretend that the set I is completely random in this high level presentation.

[6] Some additional effort is needed to make this idea go through for online channels, as after the permutation, input bits "arrive" in a way that doesn't respect the partitioning into blocks. A preliminary version of [6] contained an alternative, and more complex approach in order to deal with online channels.

The set I will define which blocks are "control blocks", and the final embedding of $c_{\text{data}}, c_{\text{ctrl}}$ into an N bit string, is done by placing c_{ctrl} in the control blocks, and c_{data} in the remaining blocks (which are suitably called data blocks). The sampling of I guarantees that for every fixed error vector e of relative hamming weight at most p, at least an $\epsilon/2$ fraction of the control blocks, are not hit with significantly more than pb errors. This will suffice for the decoding algorithm.

The decoder (that does not know I) will go over all n blocks, treating each one of them as a potential control block. Even if no errors are inflicted, only $\epsilon \cdot n$ of the n blocks are indeed control blocks. We want the decoder to be able to "list-decode" and output a small list of candidates s for the "control information".

This can be done as follows: When preparing c_{ctrl}, the control information S will be encoded by a concatenated code, where the outer code is list-decodable (or more generally list-recoverable) and has block length $\epsilon \cdot n$, and the inner code has symbols of b bits, and is decodable from slightly more than pb errors. This way, if at least $\epsilon/2$ fraction of the control blocks suffer not much more than pb errors (and are therefore decoded correctly by the inner code) then the list decoding algorithm of the outer code produces a list of candidates that includes the correct control information s. Decoding can now proceed, and for each such candidate s, it can apply Dec' on the data part (defined by s_{samp}) using the control information (s_π, s_{PRG}). This indeed suffices for list decoding against additive channels.

Extending the approach to computationally bounded channels

There is an obvious concern if we use this strategy against channels that are not additive: The channel C may inspect the different n blocks, and try to identify which of them are control blocks. It is crucial that the channel will not be able to distinguish a control block from a data block. This means that we want the inner code that produces the b-bit control blocks to have three properties:

- It should be able to decode from roughly pb errors.
- The channel should not be able to distinguish control blocks from data blocks.
- Control blocks shouldn't reveal information about S to the channel.

Here, it is useful that the data part is xored with $PRG(S_{PRG})$ and is therefore pseudorandom. This means that we can obtain these three properties if we use a stochastic code (instead of a standard code) and require that the output is pseudorandom. Note that here the notion of stochastic codes is not used to "improve decoding properties" (we can do with standard codes). Instead, it is used to perform encoding in a way that does not reveal information about the message. This notion of stochastic codes is defined in the next section.

2.4 Pseudorandom stochastic inner codes

Guruswami and Smith considered the following version of stochastic codes. Let $\text{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^b$ be a function.

1. We say that Enc is **ϵ-pseudorandom** for a class of functions $\mathcal{C}$ if for every $m \in \{0,1\}^k$ and for every $C \in \mathcal{C}$, the distribution $\text{Enc}(m, U_d)$ is ϵ-pseudorandom for C, meaning that: $|\Pr[C(\text{Enc}(m, U_d)) = 1] - \Pr[C(U_b) = 1]| \le \epsilon$.
2. We say that Enc is **L-list decodable** with **radius** p if there exists a function Dec such that for every $y \in \{0,1\}^b$, $\text{Dec}(y)$ produces a list of at most L pairs $(m, r) \in \{0,1\}^k \times \{0,1\}^d$ that contains all pairs $(m, r) \in \{0,1\}^k \times \{0,1\}^d$ such that $\delta(y, \text{Enc}(m, r)) \le p$.

Such codes can be plugged in as "inner control codes" in the scheme described in the previous section, and the two properties above suffice for the correctness of the construction (if pseudorandomness is guaranteed against a class sufficiently stronger than the channel, as explained in Section 2.2).

Consequently, the task of explicitly constructing stochastic codes against bounded channels reduces to explicitly constructing such stochastic codes with constant size lists. Here, we benefit from the fact that these codes are used as inner codes. The block length b of the inner stochastic code can be much smaller than the block length N of the final code. Note however that pseudorandomness needs to hold with respect to channels (and even more complex functions) that have complexity measured as a function of N (which in turn gives a lower bound on b).

We first concentrate on the case where channels are circuits of size N^c (which is the case considered by Guruswami and Smith). This allows setting $k, d, b = O(\log N)$ which in turn means that: we need pseudorandomness against circuits of size $N^c = 2^{\Omega(b)}$, and we are allowed to perform encoding and list-decoding in time $2^{O(b)}$.

However, even with these choices, it seems hard to construct such stochastic codes (no matter what complexity assumption we use). Guruswami and Smith [6] were not able to give such explicit constructions. Instead, they settle for a Monte-Carlo construction using the probabilistic method: They describe a probability space over functions $\text{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^b$ for $k, d, b = O(\log N)$ in which a code with the two properties above is chosen with high probability. The description of Enc in this probability space is of length polynomial in N^c, and so this indeed gives a Monte-Carlo construction.[7]

2.5 New constructions of pseudorandom weak inner codes

We observe that we can relax the second property in the definition of stochastic inner codes, and still be able to use them in the framework described in the earlier sections. Specifically, let $\text{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^b$ be a function, we use the following modification of condition (2) above:

2'. We say that Enc is ***L*-weakly list decodable** with **radius** p if there exists a function Dec such that for every $y \in \{0,1\}^b$, $\text{Dec}(y)$ produces a list of at most L messages $m \in \{0,1\}^k$ that contains all messages $m \in \{0,1\}^k$ for which there exists $r \in \{0,1\}^d$ such that $\delta(y, \text{Enc}(m,r)) \leq p$.

The key difference between "weakly list decodable" and the notion used by Guruswami and Smith (which we will call "strongly list-decodable") is that this definition allows a message m to be encoded to the same value under many different seeds r, whereas the previous definition did not. It turns out that constructing codes with properties 1 and 2′ is significantly simpler than constructing codes with the original properties. Specifically, for the case of inputs and outputs that are of length $O(\log N)$, we give a general transformation that for every $0 < p < \frac{1}{2}$ takes:

- a pseudorandom generator $G : \{0,1\}^{a' \cdot \log N} \to \{0,1\}^{q \log N}$ that is pseudorandom for $\mathcal{C}$,

and converts it into,

- a stochastic code $\text{Enc} : \{0,1\}^{a \log N} \times \{0,1\}^{a' \log N} \to \{0,1\}^{q \log N}$ that is pseudorandom for $\mathcal{C}$, and is L-weakly list decodable with radius p. Here, a, a', q are constants and q is sufficiently larger than a, a' (the exact dependence is $q \geq \frac{a+a'}{1-H(p)-1/(L+1)}$). Furthermore, encoding and list-decoding can be done in time $\text{poly}(N^q)$ times the running time of G.

[7] Note that the obvious approach to checking whether a candidate Enc is pseudorandom against circuits of size N^c requires going over all such circuits which is not feasible in polynomial time.

This transformation works by setting $\mathrm{Enc}(m,r) = E(m) \oplus G(r)$ where $E : \{0,1\}^{a \log n} \to \{0,1\}^{q \log n}$ is a random code. The argument is similar to proofs that random codes are list decodable, and explicitness is achieved by derandomizing the probabilistic argument using $(L+1)$-wise independence, and using brute force decoding. (Here it is crucial that we are allowed to encode and decode in exponential time in the input and output length).[8]

We can use these transformation to obtain stochastic codes that are weakly list-decodable from radius $0 < p < \frac{1}{2}$ and are:

- pseudorandom against size N^c circuits, using the pseudorandom generators of Impagliazzo and Wigderson [8] which rely on the assumption that there exists a constant $\beta > 0$ and a problem in $E = \mathrm{DTIME}(2^{O(n)})$ that cannot be solved by circuits of size $2^{\beta \cdot n}$ for every sufficiently large n. This gives Theorem 2.
- pseudorandom against space $O(\log n)$ ROBPs, using the pseudorandom generators of Nisan and Zuckerman [16]. This (together with the improvements explained in Section 2.2 and some additional effort that goes into making the reduction implementable by small space ROBPs, explained in Section 6.2) gives Theorem 3.

2.6 Inner stochastic codes for $\mathbf{AC}^0$

Our goal is to construct a stochastic code $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ that is weakly-list decodable from radius $p > 0$, and ϵ-pseudorandom against large circuits of constant depth d. We want these codes to have fixed $\mathrm{poly}(n)$ time encoding and decoding. This is because in the final construction, we will choose the block length n to be $N^{0.1}$ (where N is the block length of the final code). This choice will enable fooling circuits of superpolynomial size.

We will use an explicit binary linear code $\mathrm{Enc}_{AG} : \{0,1\}^{d+k} \to \{0,1\}^n$ with constant rate R that decodes pn errors. There are constructions of explicit codes with rate $R > 0$ and $p > 0$, that have the additional property that the relative distance of the dual code is at least p. Such constructions can be obtained by using the Algebraic Geometric codes of Garcia and Stichtenoth [3] (that are over constant size alphabets that can be chosen to be a power of two) and viewing them as binary codes (which preserves rate, and decreases relative distance and relative dual distance by a constant factor). A description of these codes appears in a paper by Shpilka [18] (in an appendix attributed to Guruswami), and we elaborate on this result in Section 4.3.

Let G be the $(d+k) \times n$ generator matrix of such codes, and let $G^{(t)}$ denote the $d \times n$ matrix obtained by the first d rows of G, and $G^{(b)}$ denote the bottom $k \times n$ rows of G. For simplicity let us set $k = d$, so that both are linear in n. In the construction of Garcia and Stichtenoth, it can be arranged that $G^{(t)}$ is a generator matrix for a code with similar properties, and in particular the code generated by $G^{(t)}$ has relative dual distance $p > 0$ (we may need to slightly decrease p for this to hold). We define:

$$\mathrm{Enc}(x,r) = \mathrm{Enc}_{AG}(r \circ x) = (r \circ x) \cdot G = r \cdot G^{(t)} + x \cdot G^{(b)}$$

We note that the dual code to the code defined by $G^{(t)}$ has relative distance p. This means that (the transposed of) $G^{(t)}$ is the parity check matrix of a code with relative distance p, which in turn implies that every pn columns of $G^{(t)}$ are linearly independent. This

[8] It is this argument that makes the running time of encoding/decoding of our constructions for circuits and online channels, grow with the size of the family of channels. Specifically, encoding and decoding of the inner stochastic code are done by "brute force" and in particular, require running the PRG on all seeds. The number of seeds of a PRG is typically larger than the number of potential distinguishers in the fooled class, and this means that we lose in efficiency, when we try to handle more complex channels.

gives that the distribution $r \cdot G^{(t)}$ for $r \leftarrow U_d$ is pn-wise independent, and implies that for every $x \in \{0,1\}^k$, $\mathrm{Enc}(x, U_d)$ is pn-wise independent. By Braverman's theorem [2] (see also later improvements by [21]) "polylog-wise independence fools AC^0", and in particular pn-independent distributions are pseudorandom for circuits of size $2^{n^{\Omega(1/d)}}$ and depth d.

The code Enc_{AG} is uniquely decodable from pn errors. This immediately gives that Enc it is (strongly) list-decodable with radius p.

Organization of the paper

In Section 3 we give definitions of objects used in out constructions, and the constructions from earlier work that we rely on. In Section 3.2.1 we show how to construct codes against balanced errors. In Section 4 we give precise define several variants of stochastic codes, and give constructions of inner stochastic codes that will be used in the main result. In Section 5 we present the construction of stochastic codes, and restate the theorems from the introduction in a more precise way. In Section 6 we prove the correctness of the construction (and explain how to handle weak classes of channels).

3 Ingredients used in the construction

In this section we give formal definitions of the notions and ingredients used in the construction. We also cite previous results from coding theory and pseudorandomness that are used in the construction.

3.1 Pseudorandom generators

▶ **Definition 5** (Pseudorandom generators). A distribution X on n bits is **ϵ-pseudorandom** for a class $\mathcal{C}$ of functions from n bit to one bit if for every $C \in \mathcal{C}$, $|\Pr[C(X) = 1] - \Pr[C(U_n)] = 1| \leq \epsilon$. A function $G : \{0,1\}^d \to \{0,1\}^n$ is an ϵ-**PRG** for $\mathcal{C}$ if $G(U_d)$ is ϵ-pseudorandom for $\mathcal{C}$.

In the sections below, we list the constructions of pseudorandom generators, that we use in this paper. We consider several choices of classes $\mathcal{C}$.

3.1.1 Poly-size circuits

▶ **Definition 6** (E is hard for exponential size circuits). We say that E is hard for exponential size circuits if there exists $\beta > 0$ and a language $L \in \mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ such that for every sufficiently large n, circuits of size $2^{\beta \cdot n}$ fail to compute the characteristic function of L in inputs of length n.

▶ **Theorem 7** ([8]). *If E is hard for exponential size circuits then for every constant $c > 1$, there exists a constant $b > 1$ such that for every sufficiently large n, there is a $G : \{0,1\}^{b \cdot \log n} \to \{0,1\}^n$ that is a $\frac{1}{n^c}$-PRG for circuits of size n^c. Furthermore, G is computable in time poly(n^c) (where this polynomial depends on the constant β hidden in the assumption).*

3.1.2 Oblivious read once branching program

▶ **Theorem 8** ([14, 7]). *There exist a constant $a > 1$ such that for every sufficiently large n, there is a $G : \{0,1\}^{a \cdot \log n \cdot (s + \log(1/\epsilon))} \to \{0,1\}^n$ that is ϵ-pseudorandom for ROBPs of space s. Furthermore, G is computable in time poly(n).*

We also need PRGs with error that is exponentially small in the seed length. In this setup, we only require arbitrary linear stretch.

▶ **Theorem 9** ([16]). *For every $b > 1$, there exists a constant $a > 1$ such that for every sufficiently large n, there is a $G : \{0,1\}^{a \cdot s} \to \{0,1\}^{a \cdot b \cdot s}$ that is a $\frac{1}{2^{-s}}$-PRG for ROBPs of space s. Furthermore, G is computable in time poly(s).*[9]

3.1.3 Constant depth circuits

▶ **Theorem 10** ([13, 15, 22, 21]). *There exists a constant $a > 1$ such that for every constant d, and for every sufficiently large n, there is a $G : \{0,1\}^{(\log(s/\epsilon))^{a \cdot d}} \to \{0,1\}^n$ that is an ϵ-PRG for circuits of size s and depth d. Furthermore, G is computable in time poly(n).*

We will also use Braverman's result that polylog-wise independence fools AC^0.

▶ **Theorem 11** ([2, 21]). *There exists a constant $a > 1$ such that for every sufficiently large n, every $(\log(s/\epsilon))^{a \cdot d}$-wise independent distribution on n bits is ϵ-pseudorandom for circuits of size s and depth d.*

3.2 Error-Correcting Codes

We give a nonstandard definition of error-correcting codes below. For our purposes it is more natural to define codes in terms of a pair (Enc, Dec) of encoding and decoding algorithms. Different variants are obtained by considering different tasks (decoding, list-decoding, list-recovering) of the decoding algorithms and different types of error vectors.[10]

▶ **Definition 12** (Codes). Let k, n, q be parameters and let $\mathrm{Enc} : \{0,1\}^k \to (\{0,1\}^{\log q})^n$ be a function. We say that Enc is an encoding function for a code that is:

- **decodable** from errors in E (where $E \subseteq (\{0,1\}^{\log q})^n$) if there exists a function $\mathrm{Dec} : (\{0,1\}^{\log q})^n \to \{0,1\}^k$ such that for every $m \in \{0,1\}^k$ and every $e \in E$, $\mathrm{Dec}(\mathrm{Enc}(m) \oplus e) = m$. The standard choice of E is the set of all vectors with Hamming weight t, and such codes are said to be *decodable* from t errors.
- ***L*-list-decodable** from errors in E if the function Dec above is allowed to output a list of size at most L that contains m.
- **(α, ℓ, L)-list-recoverable** if there exists a function Dec which given a list $T \subseteq \{0,1\}^{\log q}$ of size at most ℓ, outputs a list of size at most L containing all $m \in \{0,1\}^k$ such that $\Pr_{i \leftarrow [n]}[\mathrm{Enc}(m)_i \in T] \geq \alpha$.[11]
- **(α, ℓ, L)-list-recoverable from a collection** if there exists a function Dec which given n lists $T_1, \ldots, T_n \subseteq \{0,1\}^{\log q}$ of size at most ℓ, outputs a list of size at most L containing all $m \in \{0,1\}^k$ such that $\Pr_{i \leftarrow [n]}[\mathrm{Enc}(m)_i \in T_i] \geq \alpha$.

A code is **explicit** if its encoding and decoding functions are computable in time polynomial in their input and output. The rate of the code is the ratio of the message length and output length of Enc, where both lengths are measured in bits.

[9] We remark that the construction of [16] can achieve superlinear stretch at the cost of increasing the error. In our application, it is crucial to achieve error that is exponentially small in the seed length, and this is why we state the theorem in this form.

[10] Within this section we use the standard choice of letters of error-correcting codes. However, in later sections many of these letters are reserved to denote other things, and we have to use nonconventional choices.

[11] This is a less standard notion of list-recoverability, and the more standard notion referred to as "list-recoverable" is what we call "list-recoverability from a collection" in the next item.

3.2.1 Codes for balanced errors

We will make use of codes for balanced error vectors (as explained in Section 2).

▶ **Definition 13** (balanced errors)**.** A string $e \in \{0,1\}^n$ is **(b, p, γ)-balanced** if when viewing it as $e \in (\{0,1\}^b)^{n/b}$ at most a γ-fraction of the n/b blocks have hamming weight larger than $p \cdot b$.

It is not hard to construct codes for balanced errors with rate approaching $1 - H(p)$, using code concatenation. The proof of Theorem 14 appears in Section 7.

▶ **Theorem 14** (codes against balanced errors)**.** *For every constants $0 < p < 1/2$, $\epsilon > 0$, and $\gamma \geq \epsilon$ there are constants b and $L = poly(1/\epsilon)$ such that for every sufficiently large n, there is a code (Enc, Dec) with rate $1 - H(p) - \epsilon$ that is L-list decodable against (b, p, ϵ)-balanced strings of length n. Moreover the code is explicit (encoding and list-decoding can be performed in time $poly(n)$).*

3.2.2 List recoverable codes

We will make use of the following list recoverable code.

▶ **Theorem 15** (List-recoverable codes, [20, 5])**.** *There is a constant $\beta > 0$ such that for every constants $\alpha > 0$ and $L > 1$, and every sufficiently large n, there is a code (Enc, Dec) that is $(\alpha, \beta \cdot \alpha \cdot L \cdot n, L)$-list recoverable, has rate $R \geq \frac{\beta \cdot \alpha}{L}$, and alphabet size $q = n^2$.*

This follows as Sudan [20] (see also Guruswami and Sudan [5]) showed that Reed-Solomon codes are list-recoverable from a collection. Given a code Enc that is list-recoverable from a collection, $\mathrm{Enc}'(x)_i = (\mathrm{Enc}(x), i)$ gives a code that is list recoverable, while increasing the alphabet size. This is why we have the alphabet size of $q = n^2$ (and not $q = n$) for a Reed Solomon code. This idea is also implicitly used by Guruswami and Smith [6].

3.3 Averaging Samplers

The reader is referred to Goldreich's survey [4] on averaging samplers.

▶ **Definition 16** (Averaging Samplers)**.** A function $\mathrm{Samp} : \{0,1\}^n \to (\{0,1\}^m)^t$ is an **(ϵ, δ)-Sampler** if for every $f : \{0,1\}^m \to [0,1]$,

$$\Pr_{(z_1,\ldots,z_t)\leftarrow \mathrm{Samp}(U_n)}[|\frac{1}{t}\sum_{i\in[t]} f(z_i) - \frac{1}{2^m}\sum_{x\in\{0,1\}^m} f(x)| > \epsilon] \leq \delta\,.$$

A sampler has **distinct samples** if for every $x \in \{0,1\}^n$, the elements in $\mathrm{Samp}(x)$ are distinct.

The next theorem follows from the "expander sampler". This particular form can be found (for example) in [23].

▶ **Theorem 17.** *For every sufficiently large m and every $\epsilon \geq \delta > 0$ there is a (ϵ, δ)-sampler, $Samp : \{0,1\}^{O(m+\log(1/\delta)\cdot poly(1/\epsilon))} \to (\{0,1\}^m)^t$ for any $t \geq poly(1/\epsilon) \cdot \log(1/\delta)$. Furthermore, Samp is computable in time $poly(m, 1/\epsilon, \log(1/\delta))$.*

3.4 Almost t-wise permutations

We also need the following notion of almost t-wise permutations.

▶ **Definition 18** (Almost t-wise independent permutations). A function $\pi : \{0,1\}^d \times [n] \to [n]$ is an (ϵ, t)-wise independent permutation if:

- For every $s \in \{0,1\}^d$, the function $\pi_s(x) = \pi(s,x)$ is a permutation over $[n]$.
- For every $x_1, \ldots, x_t \in [n]$, the random variable $R = (R_1, \ldots, R_t)$ defined by $R_i = \pi(s, x_i) : s \leftarrow U_d$, is ϵ-close to t uniform samples without repetition from $[n]$.

▶ **Theorem 19** ([9]). *For every t and every sufficiently large n, there exists an (ϵ, t)-wise independent permutation with $d = O(t \cdot \log n + \log(1/\epsilon))$. Furthermore, π is computable in polynomial time.*

4 Inner Stochastic codes

As explained in Sections 2.4 and 2.5, the construction will rely on an "inner stochastic code". We now give a formal definition of the properties required from these codes. This definition formalizes the looser description given in Section 2.

▶ **Definition 20.** Let k, n, q be parameters and let $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ be a function. We say that Enc is an encoding function for a stochastic code that is:

- ***L*-weakly list-decodable** with **radius** p if there exists a function Dec such that for every $y \in \{0,1\}^n$, $\mathrm{Dec}(y)$ produces a list of at most L messages that contains all messages $m \in \{0,1\}^k$ for which there exists $r \in \{0,1\}^d$ such that $\delta(y, \mathrm{Enc}(m,r)) \le p$.
- We replace "weakly" with "**strongly**" if Dec is required to produce a list of at most L pairs (m, r) that contains all pairs $(m, r) \in \{0,1\}^k \times \{0,1\}^d$ such that $\delta(y, \mathrm{Enc}(m,r)) \le p$.
- **ϵ-pseudorandom** for a class $\mathcal{C}'$ of functions from n bits to one bit, if for every message $m \in \{0,1\}^k$, $C(m, U_d)$ is ϵ-pseudorandom for $\mathcal{C}'$.

If we do not mention whether the code is weakly or strongly list-decodable, then we mean "weakly". In the remainder of this section we give explicit constructions of "inner stochastic codes" for the various channel classes that we consider. We start with a general transformation that transforms a PRG into an inner stochastic code.

4.1 PRGs give inner stochastic codes

We give a general transformation that given a PRG with:

- A seed length that is logarithmic in the complexity of the channel.
- Sufficiently large linear stretch as a function of p.

Produces a stochastic code that:

- Inherits the logarithmic seed length and pseudorandomness properties of the PRG.
- Is able to encode a string of length logarithmic in the complexity of the channel.
- Is L-weakly list decodable from radius p where L is a constant.
- Has encoding and decoding running in time polynomial in the complexity of the channel, and the running time of the PRG.

This transformation is formally stated in the next theorem. We need the following definition that formally defines the action (which we call "xored-restriction") of restricting functions to a subset of the input, and negating some of the remaining input bits. The complexity classes that we consider in this paper (AC^0, P/poly, logspace ROBPs) are all closed under xored restriction. (This is also the case for any natural nonuniform complexity class).

▶ **Definition 21** (xored restriction)**.** We say that a function C' over n' bits is an xored-restriction of a function C over n bits if there exist strings $y \in \{0,1\}^{n'}$, $a \in \{0,1\}^{n-n'}$ and a set $S \subseteq [n]$ of size n' such that for every input x', $C'(x') = C(x)$, where x is an n bit string obtained by "filling" the indices in S with $x' \oplus y$, and the indices outside of S with a.

▶ **Theorem 22** (inner stochastic code from PRG)**.** *Let $\mathcal{C}, \mathcal{C}'$ be classes of functions, and $a > 0$, $b > 0$, $L \geq 1$ and $0 \leq p < \frac{1}{2}$ be constants such that $(1 - \frac{1}{L+1}) > H(p)$, and assume that n is sufficiently large. Let $G : \{0,1\}^{b \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ be an ϵ-PRG for class $\mathcal{C}'$ such that $q \geq \frac{a+b}{1-H(p)-\frac{1}{L+1}}$. There is a stochastic code (Enc_{SC}, Dec_{SC}) where $Enc_{SC} : \{0,1\}^{a \cdot \log n} \times \{0,1\}^{b \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ that is:*

- *L-weakly list decodable from radius p.*
- *If every xored restriction of $\mathcal{C}$ is in $\mathcal{C}'$ then Enc_{SC} is ϵ-pseudorandom for $\mathcal{C}$.*
- *The algorithms Enc_{SC}, Dec_{SC} are computable in time poly$(n^{q \cdot L})$ given oracle access to G. (In particular, the code is explicit if G runs in time poly(n)).*

Proof. The code will be a combination of two functions, $E : \{0,1\}^{a \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ and $G : \{0,1\}^{b \cdot \log n} \to \{0,1\}^{q \cdot \log n}$, and we will have: $\mathrm{Enc}_{SC}(x,r) = E(x) \oplus G(r)$.

We will use a probabilistic construction (similar to that used to show existence of capacity achieving, binary list decodable codes) which we later derandomize using $(L+1)$-wise independence.

▶ **Claim 23.** *Let $E : \{0,1\}^{a \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ be chosen at random, so that the random variables $(E(x))_{x \in \{0,1\}^{a \cdot \log n}}$ are $(L+1)$-wise independent. Then, with positive probability, $Enc_{SC}(x,r) = E(x) \oplus G(r)$ is L-weakly list decodable from radius p.*

Proof of claim. Given $y \in \{0,1\}^{q \log n}$, we use $B(y,p)$ to denote the ball of radius $p \cdot (q \cdot \log n)$ centered at $y \in \{0,1\}^{q \log n}$. For every $x \in \{0,1\}^{a \log n}$ and $y \in \{0,1\}^{q \log n}$ we define a random variable indicator

$$Z^{x,y} = \begin{cases} 1 & \text{if } \exists r \in \{0,1\}^{b \cdot \log n} \text{ such that, } \mathrm{Enc}_{SC}(x,r) \in \ B(y,p) \\ 0 & \text{otherwise} \end{cases}$$

We have that:

$$\begin{aligned} \Pr[Z^{x,y} = 1] &\leq 2^{b \log n} \cdot \frac{2^{H(p) \cdot q \cdot \log n}}{2^{q \log n}} \\ &\leq 2^{\log n \cdot (b + q(H(p)-1))} \end{aligned}$$

Given a tuple $x_1, \ldots, x_{L+1} \in \{0,1\}^{a \log n}$ and $y \in \{0,1\}^{q \log n}$, let $B^{x_1,\ldots,x_{L+1},y}$ be the "bad event" that the $L+1$ points $x_1, \ldots, x_{L+1}$ all have seeds of G that make them land in the ball of y, namely:

$$B^{x_1,\ldots,x_{L+1},y} = \left\{ \forall i \in [L+1], \exists r \in \{0,1\}^{b \log n} \text{ such that } E(x_i) \oplus G(r) \in \ B(y,p) \right\}.$$

The random variables $E(x_1), \ldots, E(x_{L+1})$ are independent, and therefore,

$$\Pr[B^{x_1,\ldots,x_{L+1},y}] = \prod_{i=1}^{L+1} \Pr[Z^{x,y} = 1] \leq 2^{(\log n) \cdot (b + q(H(p)-1))(L+1)}.$$

Note that $\mathrm{Enc}_{SC}(x,r) = E(x) \oplus G(r)$ is L-weakly list decodable from radius p, if and only if E does not belong to $B^{x_1,\ldots,x_{L+1},y}$ for all choices of $x_1, \ldots, x_{L+1} \in \{0,1\}^{a \cdot \log n}$ and

$y \in \{0,1\}^{q \cdot \log n}$. Therefore, by a union bound, the probability that we don't obtain an L-weakly list decodable code from radius p, is at most:

$$\sum_{x_1,\ldots,x_{L+1},y} \Pr[B^{x_1,\ldots,x_{L+1},y}] \leq 2^{q \log n} \cdot \binom{2^{a \log n}}{L+1} \cdot 2^{(\log n)\cdot(b+q(H(p)-1))(L+1)}$$
$$< 2^{(\log n)\cdot(q+a(L+1)+(b+q(H(p)-1))(L+1))}$$

Thus, if $q \geq \frac{a+b}{1-\frac{1}{L+1}-H(p)}$, then the probability is less than one, and there exists an L-weakly list decodable code from radius p.[12] ◀

Given oracle access to a candidate function $E : \{0,1\}^{a \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ and to $G : \{0,1\}^{b \log n} \to \{0,1\}^{\log q}$ we can check whether E induces a code with the required properties in time $\mathrm{poly}(n^q)$.

It is standard that there are constructions of $2^{a \log n} = n^a$ random variables that are $(L+1)$-wise, and each variable is uniform over $\{0,1\}^{q \log n}$, that can be sampled using only $(L+1) \cdot q \log n$ random bits. Therefore, in time $\mathrm{poly}(n^{L \cdot q})$ we can go over all candidate E's, and find one which induces an L-weakly list decodable from radius p.

Once we find a good function E we are guaranteed that Enc_{SC} is ϵ-pseudorandom for $\mathcal{C}$.

▶ **Claim 24.** *For every $E : \{0,1\}^{a \cdot \log n} \to \{0,1\}^{q \cdot \log n}$, the function $Enc_{SC}(x,r) = E(x) \oplus G(r)$ is ϵ-pseudorandom for $\mathcal{C}$.*

Proof. Otherwise, there exists $x' \in \{0,1\}^{a \log n}$ and a function $C \in \mathcal{C}$ that distinguishes $\mathrm{Enc}_{SC}(x', U_{b \log n}) = E(x') \oplus G(U_{b \log n})$ from uniform. This means that there is an xored restriction C' of C that distinguishes $G(U_{b \log n})$ from uniform, and this is a contradiction. ◀

Finally, it remains to justify the claim about the decoding procedure. Given a string $y \in \{0,1\}^{q \log n}$, the decoding algorithm will use brute force to go over all $(x,r) \in \{0,1\}^{a \cdot \log n} \times \{0,1\}^{b \cdot \log n}$, and check for each whether $\delta(\mathrm{Enc}(x,r), y) \leq p$. By the L-weakly list decodable property, there will be at most L distinct values of x. The decoding complexity is $O(2^{a \log n} \cdot 2^{b \log n}) = \mathrm{poly}(n^{a+b})$ with oracle access to G. ◀

4.2 Inner Stochastic codes for circuits and ROBPs

By plugging in the pseudorandom generators from Theorems 7 and Theorem 9 in Theorem 22. We immediately obtain the following stochastic codes (that will be used in the construction).

▶ **Theorem 25** (inner stochastic code for poly-size circuits). *If E is hard for exponential size circuits then for every constant $0 \leq p < \frac{1}{2}$, $c > 1$ and $a > 0$ there exist constants L, b, q such that for every sufficiently large n, there is a stochastic code (Enc, Dec) where $Enc : \{0,1\}^{a \cdot \log n} \times \{0,1\}^{b \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ is:*

- *L-weakly list decodable from radius p.*
- *$\frac{1}{n^c}$-pseudorandom for size n^c circuits.*

Furthermore, the code is explicit. Specifically, Enc, Dec are computable in time $\mathrm{poly}(n^c)$, where the polynomial depends on p, a and the constant $\beta > 0$ hidden in the hardness assumption.

[12] We remark that it is also possible to extend proofs that random linear codes achieve list decoding capacity to show that we can obtain a linear code E that yields a good code Enc_{SC}.

▶ **Theorem 26** (inner stochastic code for online channels). *For every constant $0 \le p < \frac{1}{2}$, $c > 1$ and $a > 0$ there exist constants L, b, q such that for every sufficiently large n, there is a stochastic code (Enc, Dec) where $Enc : \{0,1\}^{a \cdot \log n} \times \{0,1\}^{b \cdot \log n} \to \{0,1\}^{q \cdot \log n}$ is:*

- *L-weakly list decodable from radius p.*
- *$\frac{1}{n^c}$-pseudorandom for space $c \log n$ ROBPs.*

Furthermore, the code is explicit. Specifically, Enc, Dec are computable in time poly(n^c) where the polynomial depends on p, a.

4.3 Inner stochastic codes for AC^0 channels

In this section we give a construction of inner stochastic codes for circuits of constant depth. This construction has the advantage that the encoding and decoding of the inner stochastic code run in fixed polynomial time, and do not depend on the size or depth of the circuit family.

▶ **Theorem 27** (inner stochastic code for AC^0). *There exist constants $p > 0$, $R > 0$ and $a > 1$ such that for every sufficiently large n, there is a stochastic code (Enc, Dec) where $Enc : \{0,1\}^{Rn} \times \{0,1\}^{Rn} \to \{0,1\}^n$ that is:*

- *1-strongly list decodable from radius p.*
- *$2^{-n^{\frac{1}{ad}}}$-pseudorandom for circuits of size $2^{n^{\frac{1}{ad}}}$ and depth d.*

Furthermore, the code is explicit. Specifically, Enc, Dec are computable in time poly(n), for a fixed universal polynomial (only the choice of what n is sufficiently large depends on the constants).

Proof. The theorem will follow from the following claim.

▶ **Claim 28.** *There exist constants $p > 0$, $R > 0$ such that for every sufficiently large n, there is a $2Rn \times n$ matrix $G^{(n)}$ such that:*

- *$G^{(n)}$ is a generator matrix for a binary linear $[n, 2Rn]$-code that is decodable from pn errors.*
- *Let $G_t^{(n)}$ be the $Rn \times n$ matrix obtained by taking the first Rn rows of $G^{(n)}$. $G_t^{(n)}$ is a generator matrix for a binary linear $[n, Rn]$-code such that its dual code has distance larger than pn.*
- *The code (Enc, Dec) that is defined by $G^{(n)}$ is explicit (and in particular $G^{(n)}$ can be constructed in time poly(n)).*

Proof of claim. It is sufficient to prove the lemma for codes with alphabet size 2^s for some constant s (rather than for binary codes). This is because, such codes can be viewed as binary codes (in a natural way) and this viewpoint preserves rate, and decreases relative distances (or the fraction of errors that can be decoded) by a constant factor of $1/s$. We therefore focus on proving the claim for codes with alphabet size that is constant and a power of two.

There are codes (based on algebraic geometric codes) over constant size alphabet where the size can be a power of two, that have: constant rate, can be explicitly encoded and decoded from a constant fraction of errors, and furthermore have a positive relative dual distance. Such codes follow from the work of Garcia and Stichtenoth [3] and a self contained summary is presented in [18] (the summary is in an appendix written by Guruswami). Theorem 24 in the appendix contains a precise statement on the existence of such codes.

An inspection of the proof reveals that this argument can also be used to obtain two explicit linear codes $C_t \subseteq C$ with the properties above. More specifically, by varying the parameters in the proof, there exist constants $R > 0$ and $p > 0$ such that for sufficiently large n, C_t has constant rate $R > 0$, C has rate $2R > 0$ and both codes have the properties listed above, namely: C_t (resp. C) can be efficiently decoded from $p \cdot n$ errors (for some $p > 0$) and both codes have dual distance $p \cdot n$. Loosely speaking, this follows as one can perform the argument once to obtain one code C_t, and then increase the dimension, to give a code C such that $C_t \subseteq C$ with the same properties.

The matrix G_t will be the generator matrix of C_t and it can be easily extended to a generator matrix G of C. ◀

We now observe that the claim implies the theorem. The stochastic code $Enc : \{0,1\}^{Rn} \times \{0,1\}^{Rn} \to \{0,1\}^n$ is defined as follows: Given $x, r \in \{0,1\}^{Rn}$, let y be the concatenation $y = r \circ x$ and $Enc(x,r) = y \cdot G$.

This code is 1-strongly list decodable from radius p by the decoding properties of the code generated by G. More precisely, given $z \in \{0,1\}^n$, we can decode to a unique message $y \in \{0,1\}^{2Rn}$ that has hamming distance at most pn from z, and this message $y = (x,r)$ can be found efficiently.

We now show the pseudorandomness of Enc. Let G_b denote the bottom Rn rows of G (and recall that G_t denotes the top Rn rows of G). For every $x, r \in \{0,1\}^{Rn}$,

$$\mathrm{Enc}(x,r) = (r \circ x) \cdot G = r \cdot G_t + x \cdot G_b.$$

The generator matrix G_t generates a code with dual distance at least pn. This means that transposed matrix is the parity matrix of the dual code. The fact that the dual code has distance larger than pn, implies that every pn rows of G_t are linearly independent. This gives that the distribution $r \cdot G_t$ for $r \leftarrow U_{Rn}$ is pn-wise independent, and implies that for every $x \in \{0,1\}^{Rn}$, $\mathrm{Enc}(x, U_{Rn})$ is pn-wise independent. Braverman [2] (and later improvements by Tal [21]) (See Theorem 11) showed that t-wise independent distributions are ϵ-pseudorandom for circuits of size s and depth d, if $t \geq (\log \frac{s}{\epsilon})^{c \cdot d}$ for some constant c. This gives that there exists a constant $a > 1$ such that $\mathrm{Enc}(x, U_{Rn})$ is $2^{-n^{\frac{1}{ad}}}$-pseudorandom for circuits of size $2^{n^{\frac{1}{ad}}}$ and depth d, as required. ◀

5 The construction of stochastic codes

In this section we give the construction of the stochastic code. Our construction imitates that of Guruswami and Smith [6] (with the modifications explained in Section 2). We start with introducing some notation.

Partitioning codewords into control blocks and data blocks

The construction will think of codewords $c \in \{0,1\}^N$ as being composed of $n = n_{\text{ctrl}} + n_{\text{data}}$ blocks of length $b = N/n$. Given a subset $I \subseteq [n]$ of n_{ctrl} distinct indices, we can decompose c into its data part $c_{\text{data}} \in \{0,1\}^{N_{\text{data}} = n_{\text{data}} \cdot b}$ and its control part $c_{\text{ctrl}} \in \{0,1\}^{N_{\text{ctrl}} = n_{\text{ctrl}} \cdot b}$. Similarly, given strings c_{data} and c_{ctrl} we can prepare the codeword c (which we denote by $(c_{\text{data}}, c_{\text{ctrl}})^I$ by the reverse operation. This is stated formally in the definition below.

▶ **Definition 29.** Let $I = \{i_1, \ldots, i_{n_{\text{ctrl}}}\} \subseteq [n]$ be a subset of indices of size n_{ctrl}.

- Given strings $c_{\text{data}} \in \{0,1\}^{N_{\text{data}}}$ and $c_{\text{ctrl}} \in \{0,1\}^{N_{\text{ctrl}}}$ we define an N bit string c denoted by $(c_{\text{data}}, c_{\text{ctrl}})^I$ as follows: We think of $c_{\text{data}}, c_{\text{ctrl}}, c$ as being composed of blocks of length b (that is $c_{\text{data}} \in (\{0,1\}^b)^{n_{\text{data}}}$, $c_{\text{ctrl}} \in (\{0,1\}^b)^{n_{\text{ctrl}}}$ and $c \in (\{0,1\}^b)^n$). We enumerate the indices in $[n] \setminus I$ by $j_1, \ldots, j_{n_{\text{data}}}$ and set $c_\ell = \begin{cases} (c_{\text{ctrl}})_k & \text{if } \ell = i_k \text{ for some } k; \\ (c_{\text{data}})_k & \text{if } \ell = j_k \text{ for some } k \end{cases}$

Parameters:

- N – The length (in bits) of the codeword. (Throughout we assume that N is sufficiently large). Other parameters are either constants or chosen as a function of N.
- p – The fraction of errors we need to recover from. This is a constant.
- $\mathcal{C}'$ – A class of functions (typically slightly stronger than the class of channels we allow).
- $0 < \epsilon < \frac{1}{2} - p$ – We want rate $R = 1 - H(p) - \epsilon$, meaning that messages have length RN. ϵ is a constant.
- b – We will divide the N output bits to $n = N/b$ blocks of length b, where $2\log N \leq b \leq N^{1/10}$ is a function of N that will be chosen later on. This implies $n \geq N^{0.9}$.
- $\nu \geq 2^{-\sqrt{N}}$ – A bound on the failure probability of decoding (can be chosen as a function of N).

Internal parameters:

- Blocks will be of two kinds: "control" and "data". We set $n_{\text{ctrl}} = \epsilon \cdot n$ and $n_{\text{data}} = n - n_{\text{ctrl}}$ so that $n = n_{\text{ctrl}} + n_{\text{data}}$. Let $N_{\text{ctrl}} = b \cdot n_{\text{ctrl}}$ and $N_{\text{data}} = b \cdot n_{\text{data}}$. So that $N = N_{\text{ctrl}} + N_{\text{data}}$.
- Let $\alpha > 0$ be a sufficiently small constant that will be chosen later.
- Let $\ell_{\text{ctrl}} = N^{0.8}$ and $\ell'_{\text{ctrl}} = \ell_{\text{ctrl}}/3$.

Ingredients that depend on the choice of channel class: We assume that we are given:

- A stochastic code $\text{Enc}_{SC} : \{0,1\}^{2\log n_{\text{ctrl}}} \times \{0,1\}^{\ell'_{\text{SC}}} \to \{0,1\}^b$ that is ϵ_{SC}-pseudorandom for $\mathcal{C}'$ (for $\epsilon_{SC} = \frac{\nu}{10 \cdot n_{\text{ctrl}}}$) and is L_{SC}-weakly list decodable from radius $p + \epsilon$. We require that L_{SC} is a constant, and $\ell'_{\text{SC}} \leq N$.
- An ϵ_{PRG}-PRG $\text{PRG} : \{0,1\}^{\ell'_{\text{ctrl}}} \to \{0,1\}^{N_{\text{data}}}$ for $\mathcal{C}'$, for $\epsilon_{PRG} = \frac{1}{10} \cdot \nu$.

Other Ingredients:

- A code $\text{Enc}_{\text{balanced}} : \{0,1\}^{RN} \to \{0,1\}^{N_{\text{data}}}$ with an algorithm $\text{Dec}_{\text{balanced}}$ that performs L_{balanced}-list decoding from $(b', p+\alpha, \alpha)$-balanced errors. By Theorem 14 we have an explicit construction with rate $R' \geq 1 - H(p+\alpha) - \alpha$ where b' and L_{balanced} are large constants (chosen as a function of the constants α and p). By choosing a sufficiently small $\alpha > 0$ we indeed have $R' \geq RN/N_{\text{data}} = R/(1-\epsilon)$.
- A code $\text{Enc}_{LR} : \{0,1\}^{\ell_{\text{ctrl}}} \to (\{0,1\}^{2\log n_{\text{ctrl}}})^{n_{\text{ctrl}}}$ that is $(\frac{\epsilon^2}{100}, L_{SC} \cdot n, L_{LR})$-list recoverable. Note that $L_{SC} \cdot n = \frac{L_{SC}}{\epsilon} \cdot n_{\text{ctrl}}$. By Theorem 15 we can obtain such a code with constant rate $R' > 0$ for some constant L_{LR} (these two constants depend on ϵ). The rate we allow for Enc_{LR} above is $\frac{\ell_{\text{ctrl}}}{2\log n_{\text{ctrl}} \cdot n_{\text{ctrl}}} \leq \frac{N^{0.8}}{\epsilon \cdot n} = o(1) \leq R'$.
- A $(2^{-N^{0.6}}, N^{0.6})$-wise permutation $\pi : \{0,1\}^{\ell'_{\text{ctrl}}} \times [N_{\text{data}}] \to [N_{\text{data}}]$. By Theorem 19 we have an explicit construction with seed length $N^{0.7} \leq \ell'_{\text{ctrl}}$.
- An $(2^{-N^{0.6}}, min(\frac{\alpha}{100}, \frac{\epsilon^2}{100}))$-sampler with distinct samples $\text{Samp} : \{0,1\}^{\ell'_{\text{ctrl}}} \to [n]^{n_{\text{ctrl}}}$. By Theorem 17 we have an explicit construction with seed length $O(N^{0.7}) \leq \ell'_{\text{ctrl}}$ and $N^{0.7} \leq \epsilon \cdot n = n_{\text{ctrl}}$ samples.

Figure 1 Parameters and ingredients for stochastic code.

- Given a string $c \in \{0,1\}^N$ (which we think of as $c \in (\{0,1\}^b)^n$) we define strings $c^I_{\text{data}}, c^I_{\text{ctrl}}$ by $c_{\text{ctrl}} = c|_I$ and $c_{\text{data}} = c|_{[n]\setminus I}$, (namely the strings restricted to the indices in I, $[n] \setminus I$, respectively).

We omit the superscript I when it is clear from the context.

Permuting strings

Our construction will also use permutations to permute strings as follows:

▶ **Definition 30.** Given a string $v \in \{0,1\}^N$ and a permutation $\pi : [N] \to [N]$. Let $\pi(v)$ denote the string $v' \in \{0,1\}^N$ with $v'_i = v_{\pi(i)}$.

Input:

- A message $m \in \{0,1\}^{RN}$.
- A "random part" r for the stochastic encoding that consists of a string $s = (s_{\text{samp}}, s_\pi, s_{\text{PRG}})$ where $s_{\text{samp}}, s_\pi, s_{\text{PRG}} \in \{0,1\}^{\ell'_{\text{ctrl}}}$ so that $s \in \{0,1\}^{\ell_{\text{ctrl}}}$, and $r_1, \ldots, r_{n_{\text{ctrl}}} \in \{0,1\}^{\ell'_{\text{SC}}}$.

Operation:

Determine control blocks: Apply $\text{Samp}(s_{\text{samp}})$ to generate $I = \{i_1, \ldots, i_{n_{\text{ctrl}}}\} \subseteq [n]$. These blocks will be called "control blocks", and the remaining n_{data} blocks will be called "data blocks".

Prepare data part: We prepare a string c_{data} of length N_{data} as follows:

- Encode m by $x = \text{Enc}_{\text{balanced}}(m)$.
- Generate an N_{data} bit string y by reordering the N_{data} bits of the encoding using the (inverse of) the permutation $\pi_{s_\pi}(\cdot) = \pi(s_\pi, \cdot)$. More precisely, $y = \pi_{s_\pi}^{-1}(x) = \pi_{s_\pi}^{-1}(\text{Enc}_{\text{balanced}}(m))$.
- Mask y using PRG. That is, $c_{\text{data}} = y \oplus \text{PRG}(s_{\text{PRG}}) = \pi_{s_\pi}^{-1}(\text{Enc}_{\text{balanced}}(m)) \oplus \text{PRG}(s_{\text{PRG}})$.

Prepare control part: We prepare a string c_{ctrl} of length N_{ctrl} (which we view as n_{ctrl} blocks of length b) as follows:

- Encode s by $z = \text{Enc}_{LR}(s)$. This is a string composed of n_{ctrl} blocks of length $2\log n_{\text{ctrl}}$.
- Use Enc_{SC} as an "inner code" to encode blocks of z using the randomness $r_1, \ldots, r_{n_{\text{ctrl}}}$. That is, $(c_{\text{ctrl}})_j = \text{Enc}_{SC}(z_j, r_j) = \text{Enc}_{SC}(\text{Enc}_{LR}(s)_j, r_j)$.

Merge data and control parts: We prepare the final output codeword $c \in \{0,1\}^N$ by merging c_{data} and c_{ctrl}. That is, $c = (c_{\text{data}}, c_{\text{ctrl}})^I$.

Figure 2 Encoding algorithm for stochastic code.

Description of the construction

Our construction is described in detail in the three figures below. The choice of parameters and ingredients is described in Figure 1. The encoding algorithm is described in Figure 2, and the list-decoding algorithm is described in Figure 3. We state a general theorem that summarizes the correctness of the construction and will be used to prove Theorems 2, 3, 4.

Correctness of the construction

Let $\mathcal{C}$ be a class of channels $C : \{0,1\}^N \to \{0,1\}^N$ that induce at most pN errors. We now show that if the ingredients $\text{PRG}, \text{Enc}_{SC}$ are pseudorandom for a class $\mathcal{C}'$ that is sufficiently stronger than $\mathcal{C}$, then the decoding algorithm of Figure 3 succeeds with high probability. This is stated precisely, in the next theorem, which uses the notion of "xored restrictions" defined in Definition 21. (We remind the reader that nonuniform complexity classes as the ones we consider in this paper, are closed under xored restrictions).

▶ **Theorem 31** (Correctness of construction). *For every constants $0 \le p < \frac{1}{2}$ and $0 < \epsilon < \frac{1}{2} - p$ there exists a constants $L = L_{LR} \cdot L_{balanced}$ such that for every sufficiently large N the following holds:*

- *Let $\mathcal{C}$ be a class of functions $C : \{0,1\}^N \to \{0,1\}^N$ that induce at most pN errors. For a channel $C \in \mathcal{C}$, let $E_C(z) = z \oplus C(z)$ denote the error vector (of Hamming weight at most pN) induced by the channel.*
- *Let $\mathcal{C}'$ be the class of all functions that output one bits, and are xored restrictions of functions of the form $f(z) = A(E_C(z)))$ where A is either,*
  - *a size N^{c_0}, depth d_0 circuit, for some universal constants c_0, d_0.*

```
Input: A "received word" $c' \in \{0,1\}^{RN}$.
Operation:
  Determine few candidates for control information:
    Decode inner code SC: For every $i \in [n]$ apply the list decoding algorithm of $SC$ to
      generate a size $L_{SC}$ list, $\mathrm{List}_i = \mathrm{Dec}_{SC}(c'_i)$ (here $c'_i$ is the $i$'th block of $c'$). Let $\mathrm{List}_{SC} = \cup_{i \in [n]} List_i$.
    Decode outer code LR: Apply the list recovering algorithm of $LR$ to generate a size $L_{LR}$
      list, $\mathrm{List}_{\mathrm{ctrl}} = \mathrm{Dec}_{LR}(List_{SC})$.
  Use each control candidate $s$ to decode data: For each $s = (s_{\mathrm{samp}}, s_\pi, s_{\mathrm{PRG}}) \in \mathrm{List}_{\mathrm{ctrl}}$ (re-
    call that there are $L_{LR}$ of them) we produce a list $\mathrm{List}_s$ of $L_{\mathrm{balanced}}$ candidate messages.
    Our final output list will be the union of these lists.
    Determine control blocks: Apply $\mathrm{Samp}(s_{\mathrm{samp}})$ to generate $I = \{i_1, \dots, i_{n_{\mathrm{ctrl}}}\}$. Compute
      $c'_{\mathrm{data}} = (c')^I_{\mathrm{data}}$.
    Unmask PRG: Compute $y'_{\mathrm{data}} = c'_{\mathrm{data}} \oplus \mathrm{PRG}(s_{\mathrm{PRG}})$.
    Reverse permutation: Let $x'$ be the $N_{\mathrm{data}}$ bit string obtained by "undoing" the permutation.
      More precisely, let $\pi_{s_\pi}(\cdot) = \pi(s_\pi, \cdot)$, and let $x' = \pi_{s_\pi}(y'_{\mathrm{data}}) = \pi_{s_\pi}(c'_{\mathrm{data}} \oplus \mathrm{PRG}(s_{\mathrm{PRG}}))$.
    Decode data: Compute $\mathrm{List}_s = \mathrm{Dec}_{\mathrm{balanced}}(x')$.
    Merge lists: The final output is $\mathrm{List} = \bigcup_{s \in \mathrm{List}_{\mathrm{ctrl}}} \mathrm{List}_s$.
```

Figure 3 List-decoding algorithm for stochastic code.

- *a space $\eta_0 \cdot \log 1/\nu \cdot \log N$ ROBP, for some universal constant $\eta_0 > 0$ (which gives space $O(\log N)$ if ν is inverse polynomial in N).*

If the parameters and ingredients are chosen as in Figure 1, then the stochastic code (Enc, Dec) specified in Figures 2, 3, satisfies:

- *It has rate $R \geq 1 - H(p) - \epsilon$.*
- *It is L-list decodable with success probability $1 - \nu$ for channels in $\mathcal{C}$, where $L = poly(1/\epsilon)$ is a constant.*
- *There exist a universal polynomial $P(\cdot)$ such that:*
  - *The function Enc can be computed in $DTIME^{PRG, Enc_{SC}}(P(N))$ (and is therefore explicit if PRG, Enc_{SC} are explicit).*
  - *The function Dec can be computed in $DTIME^{PRG, Dec_{SC}}(P(N))$ (and is therefore explicit if PRG, Dec_{SC} are explicit).*

5.1 Choosing ingredients and parameters for specific channel families

We now put everything together and choose pseudorandom generators and inner stochastic codes for poly-size circuits, online logspace, and AC^0.

5.1.1 Poly-size circuit channels

Here we use the pseudorandom generator of Impagliazzo and Wigderson [8] (that requires the assumption that E is hard for exponential size circuits). This PRG has logarithmic seed length, and can be used as PRG, as well as the pseudorandom generator that is transformed into an inner stochastic code Enc_{SC} (as done in Theorem 25). The precise statement and parameter choices appear below:

▶ **Theorem 32** (explicit codes for poly-size channels). *Assume that E is hard for exponential size circuits. For every constants $0 \le p < \frac{1}{2}$, $\epsilon > 0$, and $c > 1$ and for every sufficiently large N:*

- *Let $\nu = N^{-c}$.*
- *Let $\mathcal{C}$ be the class of all circuits $C : \{0,1\}^N \to \{0,1\}^N$ of size N^c that induce at most pN-errors.*
- *Let $\mathcal{C}'$ be the class of all size N^{2c} circuits that output one bit (this includes circuits for all input lengths up to N). Here, we assume w.l.o.g. that c is sufficiently large so that in time N^{2c} we can compose size N^c computations with fixed polynomial size computations.*
- *Let (Enc_{SC}, Dec_{SC}) and the block length b be determined by Theorem 25. Specifically, let $b = q \cdot \log N$ for a sufficiently large constant q, guaranteed by Theorem 25 so that we get that $Enc_{SC} : \{0,1\}^{2 \log n_{ctrl} \le 2 \log N} \times \{0,1\}^{\ell'_{SC} = O(\log N)} \to \{0,1\}^b$ is L_{SC}-weakly list decodable from radius $p + \alpha$ for a sufficiently large constant L_{SC} (chosen as a function of p), and furthermore, Enc_{SC} is $N^{-(c+1)}$-pseudorandom for $\mathcal{C}'$. (Note that $N^{-(c+1)} \le \nu/10 \cdot n_{ctrl}$ as required).*
- *Let $PRG : \{0,1\}^{O(\log N)} \to \{0,1\}^{N_{data}}$ be an $N^{-(c+1)}$-PRG for $\mathcal{C}'$ from Theorem 7, and note that the seed length is smaller than ℓ'_{ctrl}, and $N^{-(c+1)} \le \nu/10$ as required.*

These choices satisfy the requirements of Figure 1, 2, 3, the stochastic code (Enc, Dec) specified in the figures has rate $1 - H(p) - \epsilon$, and is $L = O(1)$-list decodable with success probability $1 - N^{-c}$ against channels in $\mathcal{C}$. Furthermore, Enc, Dec are computable in time poly(N^c) where the polynomial depends on p, and on the constant $\beta > 0$ hidden in the assumption.

5.1.2 Online logspace channels

Here we use the pseudorandom generator of Nisan [14]. This PRG has seed length that is poly-logarithmic, and can be used as PRG. However, it unsuitable to serve in the construction of inner stochastic codes. This is because the dependence of the seed length on the error, does not allow linear stretch with error that is exponentially small in the seed length. Instead, we use the pseudorandom generator of Nisan and Zuckerman [16], that has these properties and can be transformed into an inner stochastic code Enc_{SC} (as done in Theorem 26). The precise statement and parameter choices appear below:

▶ **Theorem 33** (explicit codes for online logspace channels). *For every constants $0 \le p < \frac{1}{2}$, $\epsilon > 0$, $c > 1$ and for every sufficiently large N:*

- *Let $\nu = N^{-c}$.*
- *Let $\mathcal{C}$ be the class of all space $c \log N$ ROBPs $C : \{0,1\}^N \to \{0,1\}^N$ that induce at most pN-errors.*
- *Let $\mathcal{C}'$ be the class of all space $2c \log N$ ROBPs that output one bit (this includes ROBPs for all input lengths up to N). Here we assume w.l.o.g. that c is sufficiently large so that an ROBP of space $2c \log N$ can compose space c $logN$ online computation with $c_0 \log N$ online computation, for any fixed c_0.*
- *Let (Enc_{SC}, Dec_{SC}) and the block length b be determined by Theorem 26. Specifically, let $b = q \cdot \log N$ for a sufficiently large constant q, guaranteed by Theorem 26 so that we get that $Enc_{SC} : \{0,1\}^{2 \log n_{ctrl} \le 2 \log N} \times \{0,1\}^{\ell'_{SC} = O(\log N)} \to \{0,1\}^b$ is L_{SC}-weakly list decodable from radius $p + \alpha$ for a sufficiently large constant L_{SC} (chosen as a function of p), and furthermore, Enc_{SC} is $N^{-(c+1)}$-pseudorandom for $\mathcal{C}'$. (Note that $N^{-(c+1)} \le \nu/10 \cdot n_{ctrl}$ as required).*
- *Let $PRG : \{0,1\}^{O(\log^2 N)} \to \{0,1\}^{N_{data}}$ be an $N^{-(c+1)}$-PRG for $\mathcal{C}'$ from Theorem 8, and note that the seed length is smaller than ℓ'_{ctrl}, and $N^{-(c+1)} \le \nu/10$ as required.*

These choices satisfy the requirements of Figure 1, 2, 3, the stochastic code (Enc, Dec) *specified in the figures has rate* $1 - H(p) - \epsilon$, *and is* $L = O(1)$*-list decodable with success probability* $1 - N^{-c}$ *against channels in* $\mathcal{C}$. *Furthermore,* Enc, Dec *are computable in time* $poly(N^c)$ *where the polynomial depends on* p.

5.1.3 Constant depth channels

Here we use the pseudorandom generator of Nisan [13]. This PRG has seed length that is subpolynomial for any fixed constant depth d, and can be used as PRG. We use the construction of inner stochastic codes given in Theorem 27 for Enc_{SC}. This construction only works for $p < p_0$ for some $p_0 > 0$ and this requirement is inherited by our final theorem. The precise statement and parameter choices appear below:

▶ **Theorem 34** (explicit codes for constant depth channels). *There exists a constant* $p_0 > 0$, $d_0 > 1$ *and* $a > 0$ *such that for every constants* $0 \le p < p_0$, $\epsilon > 0$, $d > 1$ *and for every sufficiently large* N*:*

- *Let* $\nu = 2^{-N^{\frac{1}{ad}}}$.
- *Let* $\mathcal{C}$ *be the class of circuits* $C : \{0,1\}^N \to \{0,1\}^N$ *of size* $2^{N^{\frac{1}{ad}}}$ *and depth* d *that induce at most* pN*-errors.*
- *Let* $\mathcal{C}'$ *be the class of all size* $2^{2N^{\frac{1}{ad'}}}$ *and depth* $d' = d + d_0$ *circuits that output one bit (this includes circuits for all input lengths up to* N*).*
- *Let* $b = N^{1/10}$ *and let* (Enc_{SC}, Dec_{SC}) *be determined from Theorem 27. Specifically, let* $R > 0$ *be a constant guaranteed by Theorem 27 so that we get* $Enc_{SC} : \{0,1\}^{Rb} \times \{0,1\}^{Rb} \to \{0,1\}^b$ *is* L_{SC}*-weakly list decodable from radius* $p + \alpha$ *for* $L_{SC} = 1$, *and furthermore,* Enc_{SC} *is* $2^{-2N^{\frac{1}{ad}}}$*-pseudorandom for* $\mathcal{C}'$.
- *Let* $PRG : \{0,1\}^{(\log N)^{O(d')} \le Rb} \to \{0,1\}^{N_{data}}$ *be an* $2^{-2N^{\frac{1}{ad}}}$*-PRG for* $\mathcal{C}'$ *from Theorem 10, and note that the seed length is smaller than* ℓ'_{ctrl}.

These choices satisfy the requirements of Figure 1, 2, 3, the stochastic code (Enc, Dec) *specified in the figures has rate* $1 - H(p) - \epsilon$, *and is* $L = O(1)$*-list decodable with success probability* $1 - 2^{-N^{\frac{1}{ad}}}$ *against channels in* $\mathcal{C}$. *Furthermore,* Enc, Dec *are computable in time* $poly(N)$ *for a fixed universal polynomial.*

6 Analyzing the construction

This section is devoted to proving Theorem 31.

The setup

Throughout the remainder of the section, we fix the following setup: Let $0 \le p < \frac{1}{2}$ and $0 < \epsilon < \frac{1}{2} - p$ be constants. Let $\mathcal{C}, \mathcal{C}'$ be classes as required in Theorem 31. We use the choices and requirements made in Figure 1. More specifically, as in Figure 1, we assume that we are supplied with PRG and $(\mathrm{Enc}_{SC}, \mathrm{Dec}_{SC})$ that satisfy the requirements made in Figure 1. That is, that for some "required error" parameter $\nu \ge 2^{-\sqrt{N}}$ we have:

- A stochastic code $\mathrm{Enc}_{SC} : \{0,1\}^{2\log n_{\mathrm{ctrl}}} \times \{0,1\}^{\ell'_{\mathrm{SC}}} \to \{0,1\}^b$ that is ϵ_{SC}-pseudorandom for $\mathcal{C}'$ (for $\epsilon_{SC} = \frac{\nu}{10 \cdot n_{\mathrm{ctrl}}}$) and is L_{SC}-weakly list decodable from radius $p + \epsilon$, for a constant L_{SC}.
- An ϵ_{PRG}-PRG $\mathrm{PRG} : \{0,1\}^{\ell'_{\mathrm{ctrl}}} \to \{0,1\}^{N_{\mathrm{data}}}$ for $\mathcal{C}'$, for $\epsilon_{PRG} = \nu/10$.

Our goal in this section is to show that for every sufficiently large N, the encoding and decoding algorithms specified in Figures 2 and 3 satisfy the conclusion of Theorem 31. This setup is assumed throughout this section.

6.1 Milestones for correct decoding

Following Guruswami and Smith [6] we will analyze the construction in two steps: We first consider the case that the channel C is an additive channel, namely that $C(z) = z \oplus e$ for some fixed error vector e, and later extend to general channels that can choose e as a function of z.

We present the following abstraction of this method (which will be convenient for our purposes as we use several different classes of channels). We will define "milestones" (as a function of m, s_π, s_{samp} and e) and will require that:

1. If the milestones occur, then the decoding algorithm succeeds.
2. If S_π, S_{samp} are random and e is fixed (that is, if the channel is additive) then the milestones occur with probability close to one.
3. Checking whether the milestones occur is computationally easy.

We will state a general theorem showing that if such milestones exist, then the correctness of the decoding holds even against channels that are not additive, as long as the construction is using pseudorandomness against a class $\mathcal{C}'$ that can simulate the channel and milestones. This is stated formally in the definition and theorem below (in which we allow milestones to be probabilistic).

▶ **Definition 35** (Milestones function). Let $A : \{0,1\}^{RN} \times \{0,1\}^{\ell'_{\text{ctrl}}} \times \{0,1\}^{\ell'_{\text{ctrl}}} \times \{0,1\}^N \times \{0,1\}^N \to \{0,1\}$ be a function that receives as input: a message $m \in \{0,1\}^{RN}$, a sampler seed $s_{\text{samp}} \in \{0,1\}^{\ell'_{\text{ctrl}}}$, a permutation seed $s_\pi \in \{0,1\}^{\ell'_{\text{ctrl}}}$, an error vector $e \in \{0,1\}^N$ of relative hamming weight at most p, and a " random coins string" $y \in \{0,1\}^N$. We say that A is a **milestones function** (with respect to the classes $\mathcal{C}, \mathcal{C}'$) if it has all the following properties: (the probability space for the statements below is choosing the randomness of the encoder $S = (S_{\text{samp}}, S_\pi, S_{PRG})$, $R = (R_1, \ldots, R_{n_{\text{ctrl}}})$ and Y (the coins of A) uniformly and independently.)

1. For every $m \in \{0,1\}^{RN}$, $s \in \{0,1\}^{\ell_{\text{ctrl}}}, r \in (\{0,1\}^{\ell'_{\text{SC}}})^{n_{\text{ctrl}}}$ and $e \in \{0,1\}^n$ of relative hamming weight at most p, $\Pr[A(m, s_{\text{samp}}, s_\pi, e, Y) = 1] \geq \frac{1}{2} \Rightarrow m \in \text{Dec}(\text{Enc}(m, s, r) \oplus e)$.
2. For every $m \in \{0,1\}^{RN}$ and $e \in \{0,1\}^n$ of relative hamming weight at most p, $\Pr[A(m, S_{\text{samp}}, S_\pi, e, Y) = 1] \geq 1 - \nu/10$.
3. For every $m, s_{\text{samp}}, s_\pi, y, C \in \mathcal{C}$, every xored-restriction of the function $D(z) = A(m, s_{\text{samp}}, s_\pi, E_C(z), y)$ is in $\mathcal{C}'$.

▶ **Lemma 36** (Milestones Lemma). *If there exist a milestones function with respect to $\mathcal{C}, \mathcal{C}'$ then*

$$\Pr[m \in \mathit{Dec}(C(\mathit{Enc}(m, S, R)))] \geq 1 - \nu\,.$$

We defer the proof of the milestones lemma to Section 6.3. In the next section we explain how the milestones lemma implies Theorem 31.

6.2 Milestones Lemma implies Theorem 31

In this section we show that Lemma 36 implies Theorem 31. Our task is to define a milestone function that meets the three requirements in Definition 35. We start with the following definition.

▶ **Definition 37.** We say that a string $e \in \{0,1\}^N$ is (λ, η)-**good** with respect to $s_{\mathrm{samp}} \in \{0,1\}^{\ell'_{\mathrm{ctrl}}}$ if for $I = \{i_1, \ldots, i_{n_{\mathrm{ctrl}}}\} = \mathrm{Samp}(s_{\mathrm{samp}})$:

$$|\left\{j : \text{The Hamming weight of } e_{i_j} \text{ is at most } \lambda \cdot b\right\}| \geq \eta \cdot n_{\mathrm{ctrl}}.$$

We will use slightly different milestone functions for different complexity measures (as we need the milestone function to be efficient for the corresponding complexity measure). It will be convenient to start by defining two milestone functions (a strong one, and a weak one). We will later show that more efficient milestone functions can be "sandwiched" between the two milestone functions. This will mean that correctness of the more efficient milestone functions will follow by analyzing the simpler versions.

▶ **Definition 38.** It will be convenient to denote the input to a milestone function by (x, y) where $x = (m, s_{\mathrm{samp}}, s_\pi, e)$ and y is the "random coins", we define the following functions (which do not depend on y):

Control milestone: Let $\mu = \epsilon^2/4$.

- $A^{weak}_{\mathrm{ctrl}}(x, y) = 1$ iff e is $(p + \epsilon, \frac{\mu}{10})$-good for s_{samp}.
- $A^{strong}_{\mathrm{ctrl}}(x, y) = 1$ iff e is $(p + \epsilon/4, (1 - \frac{1}{10}) \cdot \mu)$-good for s_{samp}.

Note that for every (x, y), $A^{strong}_{ctrl}(x, y) = 1 \Rightarrow A^{weak}_{ctrl}(x, y) = 1$.

Data milestone: Let $\pi_{s_\pi(\cdot)} = \pi(s_\pi, \cdot)$, $e_{\mathrm{data}} = e^{\mathrm{Samp}(s_{\mathrm{samp}})}_{\mathrm{data}}$, and $e^\pi = \pi_{s_\pi}(e_{\mathrm{data}})$.

- $A^{weak}_{\mathrm{data}}(x, y) = 1$ iff e^π is $(b', p + \alpha, \alpha)$-balanced.
- $A^{strong}_{\mathrm{data}}(x, y) = 1$ iff e^π is $(b', p + \alpha/4, \alpha/4)$-balanced.

Note that for every (x, y), $A^{strong}_{\mathrm{data}}(x, y) = 1 \Rightarrow A^{weak}_{\mathrm{data}}(x, y) = 1$.

Combined milestones:

- $A^{weak}(x, y) = A^{weak}_{\mathrm{ctrl}}(x, y) \wedge A^{weak}_{\mathrm{data}}(x, y)$.
- $A^{strong}(x, y) = A^{strong}_{\mathrm{ctrl}}(x, y) \wedge A^{strong}_{\mathrm{data}}(x, y)$.

Note that for every (x, y), $A^{strong}(x, y) = 1 \Rightarrow A^{weak}(x, y) = 1$.

The next two lemmata give that any milestone function that is "sandwiched" between A^{weak} and A^{strong} satisfy the first two properties of a milestone function.

▶ **Lemma 39.** *The function A^{weak} satisfies the first property of a milestone function. (This in particular implies that A^{strong} also satisfies the first property).*

This follows as the function A^{weak} was defined precisely so that the decoding components, in the decoding algorithm of Figure 3 are used with the correct guarantee. A full proof appears in Section 6.4.

▶ **Lemma 40.** *The function A^{strong} satisfies the second property of a milestone function. (This in particular implies that A^{weak} also satisfies the second property).*

This follows as the function A^{strong} was defined precisely so that the pseudorandom components (the sampler and permutation) are "sufficiently random" to imply that A^{strong} holds. For this, we only need to analyze the case where e is fixed and the Seeds $(S_{\mathrm{samp}}, S_\pi)$ are chosen at random. A full proof appears in Section 6.5.

Milestones for poly-size circuits

Both functions A^{weak}, A^{strong} satisfy the first two properties, and are obviously computable in polynomial time. This immediately gives that they satisfy the third and final property if $\mathcal{C}'$ is sufficiently stronger than $\mathcal{C}$ in the sense that it can run poly-time computations "on top of" computations in $\mathcal{C}$. This also immediately implies Theorem 31 for the case where A is allowed to run in some fixed polynomial time.

We would like to give tighter reductions in which the milestone function is computable in AC^0 or by a small space ROBP. We now explain how to achieve such milestone functions.

Milestone function for constant depth circuits

We would like to implement the milestone function A^{weak} (or A^{strong}) by a poly-size constant depth circuit. Note that the third property in Definition 35 considers the case that $S_{\mathrm{samp}}, S_{\pi}$ are fixed to some values $s_{\mathrm{samp}}, s_{\pi}$, and the only live input is e. This means that the choice of permutation, and which blocks are control blocks is fixed (and can be hardwired as nonuniform advice) to the circuit. Furthermore, in the data milestone the inputs can be rearranged according to $\pi_{s_{\pi}}$, at no cost. Meaning that the circuit can compute e^{π} from e at no cost. Thus, computing the milestone function reduces to several counting tasks on the number of ones in e and e^{π}.

It is known that the problem of counting the number of ones in an n bit input, cannot be solved by poly-size depth circuits. However, Ajtai [1] showed that for every $\eta > 0$, there is a polynomial size constant depth circuit that can produce a quantity that is the number of ones, up to an error of ηn. (In fact, the results of Ajtai are much stronger, and in particular allow subconstant η). This means that there is a circuit with constant depth and polynomial size $A'_{s_{\mathrm{samp}},s_{\pi}}(e)$ such that for every m, y:

$$A^{strong}(m, s_{\mathrm{samp}}, s_{\pi}, e, y) = 1 \Rightarrow A'_{s_{\mathrm{samp}},s_{\pi}}(e) = 1 \Rightarrow A^{weak}(m, s_{\mathrm{samp}}, s_{\pi}, e, y) = 1\,.$$

This means that the milestone function $A^{middle}(x, y) = A'_{s_{\mathrm{samp}},s_{\pi}}(e)$ satisfies the three properties of a milestone function proving Theorem 31 for the case of constant depth circuits.

Milestones for read once branching programs

As in the case of constant depth circuits, we need to implement the milestone function by an $O(\log n)$ space ROBP for fixed $s_{\mathrm{samp}}, s_{\pi}$. Using the approach we used for constant depth circuits, this may seem easy at first glance, as ROBPs with space $O(\log n)$ can count up to $n^{O(1)}$ and this sufficed for the earlier implementation. Indeed, this reasoning applies to the control milestone, and the functions $A^{strong}_{\mathrm{ctrl}}$ and A^{weak}_{ctrl} can be easily implemented by an ROBP of space $O(\log n)$ (for fixed $s_{\mathrm{samp}}, s_{\pi}$).

The functions $A^{strong}_{\mathrm{data}}$ and A^{weak}_{data} pose a problem. Unlike circuits, an ROBP is not allowed to reorder the input by a fixed permutation $\pi_{s_{\pi}}$ prior to reading it. Thus, we cannot assume that online access to e, gives online access to e^{π}.

We do have that s_{π} is fixed, and can be hardwired to the ROBP. This means that when an ROBP reads the i'th bit of the input e, it can tell whether this bit belongs to a control block or a data block, and in the latter case, it can tell to which of the N_{data}/b' blocks of length b', does i belong to. (All these are operations that do not depend on e, and only depend on the fixed $s_{\mathrm{samp}}, s_{\pi}$). The issue is that the order in which the ROBP reads the data bits is permuted, and does not respect their partitioning into blocks of length b'. This means that the ROBP cannot keep a single counter and use it for all blocks, and must maintain ℓ different counters, if it wants to count the number of ones in ℓ different blocks. The naive way to check if e^{π} is balanced, is to keep counters for all $\ell = N_{\mathrm{data}}/b'$ blocks, and as b' is constant, this takes space $O(\ell) = O(N_{\mathrm{data}}/b')$ which is way too much.

The solution is to use randomization. The milestone function is allowed to toss random coins (in the form of the input y). It will choose $\ell = O(\log N)$ uniform indices from $[N_{\mathrm{data}}/b]$, and will only keep count of the number of ones in these blocks. (This can indeed be done in space $O(\log N)$). The milestone function will count the fraction of sampled blocks which have hamming weight larger than $p + \alpha/4$, and use this quantity ρ' as an approximation for the real quantity ρ (which is the fraction of blocks in e^{π} which have hamming weight larger than $p + \alpha/4$). By a Chernoff bound, with probability $1 - 2^{-\Omega(\alpha^2 \cdot \ell)} = 1 - N^{O(1)}$, we have that $|\rho - \rho'| \leq \alpha/100$. Therefore, the ROBP can safely output one if $\rho' \leq \alpha/2$, as this indeed implies that

$$A^{strong}_{data}(x, \cdot) = 1 \Rightarrow \Pr_Y[A^{middle}_{\text{data}}(x, Y) = 1] \geq 1 - 2^{-\Omega(\alpha^2 \ell)}$$

$$\Rightarrow \Pr_Y[A^{middle}_{\text{data}}(x, Y) = 1] \geq \frac{1}{2} \Rightarrow A^{weak}_{\text{data}}(x, \cdot) = 1.$$

This gives that by Lemma 39, A^{middle} satisfies the first property of a milestone function. By Lemma 40, A^{middle} defined in this form, satisfies the second property of milestone functions, where we suffer an additive loss of $2^{-\Omega(\alpha^2 \ell)}$ relative to what we can get for A^{strong}, because of the error induced by the Chernoff bound.

In Theorem 31, we are allowed to use space $O(\log N)$ for $\nu = 2^{-\Omega(\log N)}$, and as α is a constant, the Theorem follows.

6.3 Proof of Milestones Lemma

We prove the milestones lemma in two steps, described in the two sections below.

6.3.1 The hiding lemma

The following lemma states that for a function D that is slightly weaker than functions in $\mathcal{C}'$, an encoding of a message m is pseudorandom for D. (We will later consider the case where D is a composition of a channel and milestone functions).

▶ **Lemma 41** (Hiding Lemma). *Let D be a function such that every xored-restriction of D is in $\mathcal{C}'$. For every message $m \in \{0,1\}^{RN}$, sampler seed $s_{samp} \in \{0,1\}^{\ell'_{ctrl}}$ and permutation seed $s_\pi \in \{0,1\}^{\ell'_{ctrl}}$, let $V = Enc(m, s_\pi, s_{samp}, S_{PRG}, R_1, \cdots, R_{n_{ctrl}})$ be a random variable (defined over the probability space where $S_{PRG}, R_1, \cdots, R_{n_{ctrl}}$ are chosen uniformly and independently). It follows that V is $\frac{\nu}{5}$-pseudorandom for D, namely:*

$$|\Pr[D(V) = 1] - \Pr[D(U_N) = 1]| < \frac{\nu}{5}.$$

Proof. We assume for contradiction that there exists D such that:

$$|\Pr[D(V) = 1] - \Pr[D(U_N) = 1]| > \frac{\nu}{5}$$

and note that $\epsilon_{PRG} + n_{\text{ctrl}} \cdot \epsilon_{SC} = \nu/5$. The lemma follows from the following claim.

▶ **Claim 42.** *One of the following holds:*
- *There exists an xored-restriction C' of D such that,*
  $|\Pr[C'(PRG(S_{PRG})) = 1] - \Pr[C'(U_{N_{data}}) = 1]| > \epsilon_{PRG}$.
- *There exists $z' \in \{0,1\}^{2 \log n_{ctrl}}$ and an xored restriction C' of D, such that*
  $|\Pr[C'(Enc_{SC}(z', U_{\ell'_{SC}})) = 1] - \Pr[C'(U_b) = 1]| > \epsilon_{SC}$.

Proof of claim. We partition V into $V = (V_{\text{data}}, V_{\text{ctrl}})^{\text{Samp}(s_{\text{samp}})}$ using definition 29. We have that D distinguishes $V = (V_{\text{data}}, V_{\text{ctrl}})$ from $U_N = (U_{data}, U_{ctrl})$ with probability greater than $\nu/5$, we do a hybrid argument and consider the hybrid distribution $H = (V_{\text{data}}, U_{\text{ctrl}})$. It follows that:

- Either D distinguishes H from U_N with probability ϵ_{PRG},
- or, D distinguishes H from V with probability $n_{\text{ctrl}} \cdot \epsilon_{SC}$.

In the first case, we have that V_{data} and U_{ctrl} are independent, and an averaging argument gives that there exists a fixed value v'_{ctrl} such that D distinguishes $(U_{\text{data}}, v'_{\text{ctrl}})$ from $(V_{\text{data}}, v'_{\text{ctrl}})$ with probability ϵ_{PRG}. This gives that there exists an xored restriction of D that distinguishes U_{data} from V_{data} with probability ϵ_{PRG} and the first item of the claim holds.

In the second case, we have that m and s_π are fixed and therefore the string $y = \pi_{s_\pi}(\text{Enc}_{\text{balanced}}(m))$ used in the encoding algorithm is also fixed. The encoding algorithm computes the data part by xoring y with $PRG(S_{PRG})$ and therefore $V_{\text{data}} = \text{PRG}(S_{\text{PRG}}) \oplus y$. By an averaging argument, there exists a fixing s'_{PRG} such that D distinguishes $((PRG(s'_{PRG}) \oplus y), U_{\text{ctrl}})$ from $(((PRG(s'_{PRG}) \oplus y), V_{\text{ctrl}})|S_{PRG} = s'_{PRG})$ with probability $n_{\text{ctrl}} \cdot \epsilon_{SC}$. We have that there exists an xored restriction D' of D which distinguishes U_{ctrl} from $V'_{\text{ctrl}} = (V_{\text{ctrl}}|S_{PRG} = s'_{PRG})$.

Recall that the encoding procedure prepares the control part c_{ctrl} by preparing a string $z = \text{Enc}_{LR}(s)$ and then the j'th control block is obtained by $\text{Enc}_{SC}(z_j, r_j)$.

Having fixed $S_{PRG} = s'_{PRG}$ the only random variables that remain unfixed in V'_{ctrl} are $R_1, \ldots, R_{n_{\text{ctrl}}}$. This means that there exists a fixed z such that $(V'_{\text{ctrl}})_j = \text{Enc}_{SC}(z_j, R_j)$ and in particular, the n_{ctrl} blocks are independent. We have that D' distinguishes V'_{ctrl} from U_{ctrl} with probability $n_{\text{ctrl}} \cdot \epsilon_{SC}$, and by a standard hybrid argument, there exists an xored restriction C' of D' which distinguishes $(V'_{\text{ctrl}})_j = \text{Enc}_{SC}(z_j, R_j)$ from uniform with probability ϵ_{SC} and the second item follows. ◀

The lemma follows by the pseudorandomness properties of PRG and Enc_{SC}. ◀

6.3.2 Hiding lemma implies milestones lemma

We now show that the milestones lemma (Lemma 36) follows from the hiding lemma (Lemma 41). We are assuming that A is a milestone function with respect to $\mathcal{C}, \mathcal{C}'$ of Theorem 31. We need to show that for every message $m \in \{0,1\}^{RN}$, and every $C \in \mathcal{C}$,

$$\Pr[m \in \text{Dec}(C(\text{Enc}(m, S, R)))] \geq 1 - \nu$$

where $S = (S_{\text{samp}}, S_\pi, S_{PRG})$, $R = (R_1, \ldots, R_{n_{\text{ctrl}}})$ and Y are chosen uniformly and independently.

Fix some message $m \in \{0,1\}^{RN}$ and let $Z = \text{Enc}(m, S, R)$ denote the random variable that is the encoding of the message. We assume (for contradiction) that $\Pr[m \in \text{Dec}(C(Z))] < 1-\nu$. By the first property of a milestones function and an averaging argument we have that:

▶ **Claim 43.** $\Pr[A(m, S_{samp}, S_\pi, E_C(Z), Y) = 1] < 1 - \nu/2$.

Proof. Let $B = \{(s,r)| m \notin \text{Dec}(C(\text{Enc}(m,s,r)))\}$ be the set of pairs on which C causes a decoding error. We have that $\Pr[(S,R) \in B] \geq \nu$.

Note that for a fixed (s,r) the error vector e induced by the channel C is also fixed. We consider the probability space where $(S,R) = (s,r)$ are fixed and Y (the random coins of the function A) is chosen uniformly. By the first property of a milestone function, we have that for a fixed $(s,r) \in B$ and a fixed error e, $\Pr[A(m, s_{\text{samp}}, s_\pi, e, Y) = 0] > \frac{1}{2}$ (as otherwise decoding must succeed). Let $A' = A(m, S_{\text{samp}}, S_\pi, E_C(Z), Y)$ be the random variable of the output of function A in the probability space where S, R, Y are chosen uniformly.

$$\Pr[A' = 0] \geq \Pr[A' = 0|(S,R) \in B] \cdot \Pr[(S,R) \in B] > \nu/2$$

It follows that

$$\Pr[A(m, S_{\text{samp}}, S_\pi, E_C(Z), Y) = 1] = \Pr[A' = 1] = 1 - [A' = 0] < 1 - \nu/2. \qquad \blacktriangleleft$$

We add an independent random variable Z_U that is uniform over $\{0,1\}^N$ to our probability space (that now consists of independently chosen S, R, Y, Z_U). By the second property of a milestone function, we have that for every error vector e,

$$\Pr[A(m, S_{\text{samp}}, S_\pi, e, Y) = 1] \geq 1 - \nu/10.$$

As Z_U is independent of (S_{samp}, S_π) this holds also for an error vector of the form $E_C(Z_U)$. Namely,

$$\Pr[A(m, S_{\text{samp}}, S_\pi, E_C(Z_U), Y) = 1] \geq 1 - \nu/10.$$

This means that:

$$\Pr[A(m, S_{\text{samp}}, S_\pi, E_C(Z_U), Y) = 1] - \Pr[A(m, S_{\text{samp}}, S_\pi, E_C(Z), Y) = 1]$$
$$> (1 - \nu/10) - (1 - \nu/2) \geq \nu/4.$$

By averaging, there exist fixed values s'_{samp}, s'_π and y' such that if we consider the event $W = \{S_{\text{samp}} = s'_{\text{samp}}, S_\pi = s'_\pi, Y = y'\}$.

$$\Pr[A(m, s'_{\text{samp}}, s'_\pi, E_C(Z_U), y') = 1|W] - \Pr[A(m, s'_{\text{samp}}, s'_\pi, E_C(Z), y') = 1|W] > \nu/4.$$

We have that (S_{samp}, S_π, Y) is independent of Z_U and also independent of (S_{PRG}, R). Therefore:

$$\Pr[A(m, s'_{\text{samp}}, s'_\pi, E_C(Z_U), y') = 1] -$$
$$\Pr[A(m, s'_{\text{samp}}, s'_\pi, E_C(\text{Enc}(m, s_\pi, s_{samp}, S_{PRG}, R)), y') = 1] > \nu/4.$$

This setup (namely, where S_{samp}, S_π are fixed, and $S_{PRG}, R = (R_1, \ldots, R_{n_{\text{ctrl}}})$ are uniform) is exactly the probability space considered in the hiding lemma (Lemma 41). By the third property of milestones functions, we have that every xored restriction of the function $D(z) = A(m, s'_{\text{samp}}, s'_\pi, E_C(z), y')$ is in $\mathcal{C}'$. Therefore, the function D that we obtained gives a contradiction to the hiding lemma.

6.4 Proof of Lemma 39

We will prove the lemma in two steps that correspond to the two steps of the decoding: decoding control, and decoding data.

▶ **Claim 44.** *For every $m, s = (s_{samp}, s_\pi, s_{PRG}), r, e$ and y, let $c = Enc(m, s, r)$ and $c' = c \oplus e$. If $A^{weak}_{ctrl}(m, s_{samp}, s_\pi, e, y) = 1$ then $s \in List_{ctrl}$. Where $List_{ctrl}$ is the list obtained in the decoding algorithm described in Figure 3.*

Proof. Recall that $\text{List}_{\text{ctrl}} = Dec_{LR}(List_{SC})$, $List_{SC} = \cup_{i \in [n]} List_i$ and $\text{List}_i = \text{Dec}_{SC}(c'_i)$ (here c'_i is the i'th block of c'). By Definition 35, $A^{weak}_{\text{ctrl}}(x) = 1$ iff e is $(p + \epsilon, \frac{\mu}{10})$-good for s_{samp}. Let e_i denote the error vector restricted to the i'th block. By the properties of Enc_{SC}, if the hamming weight of e_i is less than $(p + \epsilon) \cdot b$ then $c_i \in \text{List}_i$. We have that e is $(p + \epsilon, \frac{\mu}{10})$-good for s_{samp}, and this means that for at least $\frac{\mu}{10} \cdot n_{\text{ctrl}} = \frac{\epsilon^2 \cdot n_{\text{ctrl}}}{40}$ of the n_{ctrl} control blocks $i \in I = \text{Samp}(S_{\text{samp}})$, $c_i \in \text{List}_{SC} = \cup_{i \in [n]} List_i$. Thus, we indeed have that $\Pr_{i \leftarrow [n_{\text{ctrl}}]}[\text{Enc}_{LR}(s)_i \in List_{SC}] \geq \frac{\mu}{10} > \frac{\epsilon^2}{100}$ for a set List_{SC} of size $n \cdot L_{SC}$. By the list recoverability of Enc_{LR} we get that $s \in \text{List}_{\text{ctrl}}$ meaning that the control information was successfully recovered as desired. ◀

▶ **Claim 45.** *For every $m, s = (s_{samp}, s_\pi, s_{PRG}), r, e$ and y, let $c = Enc(m, s, r)$ and $c' = c \oplus e$. If $A^{weak}_{data}(m, s_{samp}, s_\pi, e, y) = 1$ and $s \in List_{ctrl}$ (meaning that s was recovered correctly by the first step of decoding) then $m \in Dec(c')$.*

Proof. We have that $s \in \text{List}_{\text{ctrl}}$, meaning that s is one of the candidates considered in the second step of the decoding. Let y' be the string obtained from c' after the decoding uses s_{samp} to find the data blocks, s_{PRG} to unmask the data, and s_π to permute it back to it's original state. The requirement that $A^{weak}_{\text{data}}(m, s_{\text{samp}}, s_\pi, e, y) = 1$ implies that e^π is $(b', p + \alpha, \alpha)$-balanced. Note that e^π is the error vector used on the balanced code. By the guarantee on $\text{Dec}_{\text{balanced}}$ this gives that $m \in \text{List}_s = \text{Dec}_{\text{balanced}}(y')$, since the correct control is in $\text{List}_{\text{ctrl}}$ then $m \in \text{Dec}(c') = \bigcup_{s \in \text{List}_{\text{ctrl}}} \text{List}_s$ as desired. ◀

The lemma follows from the combination of both claims.

6.5 Proof of Lemma 40

A good intuition to keep in mind is that we are trying to bound the harm that can be caused by an additive channel that uses fixed error vector e of hamming weight at most p.

We start by showing that with high probability, no more than an $\epsilon^2/4$ fraction of the control blocks, suffer too many errors from the error vector e.

▶ **Claim 46.** *For every m, e of hamming weight at most pn, y, and s_π,*

$$\Pr[A^{strong}_{ctrl}(m, S_{samp}, s_\pi, e, y) = 1] \geq 1 - 2^{-N^{0.6}}.$$

Proof. For a given error vector e we define

$$T_e = \left\{i : \text{The ith block has a weight at most } (p + \tfrac{\epsilon}{4}) \cdot b\right\} .$$

For every e that has hamming weight at most pN, it holds that $|T_e| > \frac{\epsilon}{4} \cdot n$ (otherwise we would have more than pN errors). Define $f_e : [n] \to \{0, 1\}$ such that $f_e(i) = 1$ iff $i \in T_e$. By the properties of the sampler Samp,

$$\Pr_{(z_1,\ldots,z_{n_{\text{ctrl}}}) \leftarrow \text{Samp}(U_{\ell'_{\text{ctrl}}})}[|\frac{1}{n_{\text{ctrl}}}|\{i : z_i \in T_e\}| - \frac{|T_e|}{n}| > \frac{\epsilon^2}{100}] \leq 2^{-N^{0.6}}.$$

Thus, if we choose S_{samp} uniformly and independently we get that with probability $1 - 2^{-N^{0.6}}$, the number of control blocks that are good (have error less than $p + \frac{\epsilon}{4}$) is at least $(\frac{\epsilon}{4} - \frac{\epsilon^2}{100})n_{\text{ctrl}} > (\frac{9}{10} \cdot \epsilon^2/4)n_{\text{ctrl}} = ((1 - \frac{1}{10}) \cdot \mu)n_{\text{ctrl}}$. This means that the error vector e is $(p + \epsilon/4, (1 - \frac{1}{10}) \cdot \mu)$-good with probability $1 - 2^{-N^{0.6}}$ and the claim holds. ◀

We now show that the fraction of errors induced by e to the data part cannot be significantly larger than p.

▶ **Claim 47.** *For every m, e of hamming weight at most pN, y, and s_π,*

$$\Pr_{s_{samp} \leftarrow U_{\ell'_{ctrl}}}[weight(e^{Samp(s_{samp})}_{data}) \geq N_{data} \cdot (p + \frac{\alpha}{100})] \leq 2^{-N^{0.6}}$$

(here, weight is hamming weight).

Proof. For a given error vector e, we define $f_e : [n] \to [0,1]$ such that $f_e(i) = w_i$, where w_i is the relative weight of ith block in e. By the definition of the sampler

$$\Pr_{(z_1,\ldots,z_{n_{\text{ctrl}}}) \leftarrow \text{Samp}(U_{\ell'_{\text{ctrl}}})}[|\frac{1}{n_{\text{ctrl}}} \sum_{i \in [n_{\text{ctrl}}]} f(z_i) - p| > \frac{\alpha}{100}] \leq 2^{-N^{0.6}}.$$

Thus with probability $1 - 2^{-N^{0.6}}$ the number of errors induced to the control blocks is at least $N_{\text{ctrl}}(p - \frac{\alpha}{100})$, which implies that the number of error induced to the data is less than $N_{\text{data}}(p + \frac{\alpha}{100})$, and the claim follows. ◀

We will now show that permuting the data part e, produces a balanced error vector with high probability. Let s_{samp} be a sampler seed that is good with respect to the two previous claims. A $1 - 2 \cdot 2^{-N^{0.6}}$ fraction of sampler seeds, satisfy these properties. By Claim 47, we can assume that the relative hamming weight of $e_{\text{data}}^{s_{\text{samp}}}$ is at most $p + \alpha/100$. We will denote $e_{\text{data}} = e_{\text{data}}^{s_{\text{samp}}}$ in order to avoid clutter. The lemma will follow from the following claim.

▶ **Claim 48.** $\Pr[\pi(S_\pi, e_{data})$ *is* $(b', p + \alpha/4, \alpha/4)$*-balanced error*$] > 1 - e^{-\Omega(N^{0.55})}$.

This is because, together the three claims above give that with probability $1 - 2^{-N^{0.51}}$ all good events happen, and $A^{strong}(x, y) = 1$. In the remainder of this section we prove Claim 48.

Let $N' = N_{\text{data}}/b'$ be the number of b' length blocks. We now define random variables $D_1, \ldots, D_{N'}$ as follows.

$$D_i = \begin{cases} 1 & \text{The i'th block of } \pi(S_\pi, e_{\text{data}}) \text{ has weight more than } (p + \frac{\alpha}{4}) \cdot b' \\ 0 & \text{otherwise} \end{cases}$$

Claim 48 can now be seen as a claim that the sum of the D_i's is small with high probability. We will use a Chernoff style bound, due to Schmidt, Siegel and Srinivasan [17] in order to bound the probability of deviation.

▶ **Lemma 49.** *[17] Suppose $X_1, ..., X_\ell$ are binary random variables, such that for every set of distinct k indices $i_1, \cdots, i_k \in [\ell]$, $\Pr[X_{i_1} = \ldots = X_{i_k} = 1] \leq \mu^k$. If $0 < \delta \leq 1$ and $k \leq \frac{\delta \cdot \mu \cdot \ell}{2}$ then*

$$Pr[\sum_{j=1}^{\ell} X_j \geq (1 + \delta)\mu \cdot \ell] \leq e^{-\Omega(\delta k)}.$$

We plan to use Lemma 49 on the random variables $D_1, \ldots, D_{N'}$ for this purpose, we need to analyze the probability that tuples of D_i's all evaluate to one. In order to achieve this, we will first show that:

▶ **Claim 50.** *For every $v < N^{5.5}$ and every distinct $i_1, \ldots, i_v \in [N']$, and additional $i \in [N']$*

$$\Pr[D_i = 1 | D_{i_1} = \ldots = D_{i_v} = 1] \leq \alpha/10.$$

We observe that Claim 50 implies Claim 48 by Lemma 49. This is because Claim 50 implies that for $v = N^{5.5}$, and every distinct $i_1, \ldots, i_v \in [N']$,

$$\Pr[D_{i_1} = \ldots = D_{i_v} = 1] \leq (\alpha/10)^v.$$

We can now use Lemma 49 with $k = N^{5.5}$, $\delta = 1$ and $\mu = \alpha/10$ to get that:

$$Pr[\sum_{j=1}^{N'} D_j \geq \frac{\alpha \cdot N'}{5}] \leq e^{-\Omega(N^{5.5})}.$$

In order to prove Claim 50 we prove the following claim, for which we introduce the following notation: We use e^{s_π} to denote $\pi_{s_\pi}(e_{\text{data}})$. We use $e^{s_\pi}[i]$ to denote the i'th block of e^{s_π} (where blocks are of length b'). We use $e^{s_\pi}[i,j]$ to denote the j'th bit in the i'th block of e^{s_π}.

▶ **Claim 51.** *Let $v < N^{0.55}$, let $i_1, \ldots, i_v \in [N']$ be distinct blocks, let $i \in [N']$ be an additional block, and let $j_1, \ldots, j_k \in [b']$. Let $a_1, \ldots, a_v \in \{0,1\}^{b'}$ be strings such that the relative hamming weight of each a_i is at least $p + \alpha/100$. Let $E = \cap_{m \in [v]} \left\{ e^{S_\pi}[i_m] = a_m \right\}$. It follows that:*

$$\Pr[\cap_{\ell \in [k]} \left\{ e^{S_\pi}[i, j_\ell] = 1 \right\} | E] \le (p + \alpha/50)^k .$$

Proof.

$$\Pr[\cap_{\ell \in [k]} \left\{ e^{S_\pi}[i, j_\ell] = 1 \right\} | E] = \frac{\Pr[\cap_{\ell \in [k]} \left\{ e^{S_\pi}[i, j_\ell] = 1 \right\} \cap E]}{\Pr[E]}$$

Let us first imagine that π is an $(0,t)$-wise independent permutation. In this case, the denominator is some quantity $\beta \ge 1/N_{\text{data}}^v \ge 1/N^{N^{0.55}} \ge 1/2^{N^{0.56}}$ and the enumerator is at most $\beta \cdot (p + \alpha/100)^k$. This is because conditioned on the v values, the fraction of ones that is "still available" in e_{data} has not increased, and is still at most $p + \alpha/100$. It follows that the actual quantity is at most

$$\frac{\beta \cdot (p + \alpha/100)^k + 2^{-N^{0.6}}}{\beta - 2^{-N^{0.6}}} = \frac{(p + \alpha/100)^k + 2^{-N^{0.6}}/\beta}{1 - 2^{-N^{0.6}}/\beta} \le (p + \alpha/50)^k$$

where the last inequality follows for sufficiently large N because p, α and $k \le b'$ are constants, and for every two constants $A < A'$, $\frac{A + o(1)}{1 - o(1)} \le A'$. ◀

We now show that Claim 50 follows directly from Claim 51, using Lemma 49.

Proof. (of Claim 50) We use Lemma 49 on the random variables $Y_1, \ldots, Y_{b'}$ defined by:

$$Y_w = \begin{cases} 1 & e^{S_\pi}[i, w] = 1 \\ 0 & \text{otherwise} \end{cases}$$

By Claim 51 we have that for every $0 \le v < N^{5.5}$, and for every k-tuple of indices $j_1, \ldots, j_k \in [b']$ in the i'th block,

$$\Pr[Y_{j_1} = \ldots = Y_{j_k} = 1 | D_{i_1} = \ldots = D_{i_v} = 1] \le (p + \alpha/50)^k.$$

Applying Lemma 49, with $\delta = \alpha/10$, $k = \alpha^2 \cdot b'/2$, $\mu = p + \alpha/50$, and noting that $(1 + \delta) \cdot \mu \le p + \alpha/4$ we have that:

$$Pr[\sum_{j=1}^{b'} Y_j \ge (p + \alpha/4) \cdot b' | D_{i_1} = \ldots = D_{i_v} = 1] \le e^{-\Omega(\alpha^3 \cdot b')} \le \alpha/10,$$

where the last inequality follows as we are allowed to choose b' to be a sufficiently large constant as a function of α, and the claim follows. ◀

7 Proof of Theorem 14

In this section we prove Theorem 14. The high level idea is that concatenated codes easily give codes for balanced errors. A similar argument also appears in [19], for the case of codes against errors that are "t-wise independent".

Codes with the property required in Theorem 14 can be constructed by concatenating:

- An explicit outer code $C_{out} : \{0,1\}^k \to (\{0,1\}^{n_{in}})^{n_{out}}$ that is $(1-\gamma, L_{in}, L)$-list recoverable from a collection, that has rate at least $1-\epsilon/3$, and in which n_{in}, L_{in}, L are constants and $L = \text{poly}(1/\epsilon) \cdot L_{in}$.
- An inner code $C_{in} : \{0,1\}^{n_{in}} \to \{0,1\}^b$ that is L_{in}-list-decodable from $p \cdot b$ errors and has rate at least $1 - H(p) - \epsilon/3$.

Note that this indeed gives a code with the desired properties: The inner code can be list-decodable in constant time by brute force. Furthermore, for balanced error, list-decoding succeeds on $1-\gamma$ of the n_{out} blocks, giving that the list-recovering algorithm of the outer code, is set up to output a list containing the original message.

For every constant $\epsilon > 0$ if we choose sufficiently large constants n_{in}, b and $L_{in} = \text{poly}(1/\epsilon)$ then inner codes with the required property exist by a standard probabilistic argument, and as C_{in} is of constant size, we can find such codes by brute force search.

The outer code can be constructed by concatenating:

- An explicit code $C_1 : \{0,1\}^k \to (\{0,1\}^{\log n_1})^{n_1}$ that is $(1-\gamma^2, L_1, L_2 = L)$-list recoverable from a collection, and has rate at least $1-\epsilon/9$. We need that $L = \text{poly}(1/\epsilon) \cdot L_1$.
- An inner code $C_2 : \{0,1\}^{\log n_1} \to (\{0,1\}^{n_{in}})^{n_2}$ that is $(1-\gamma^2, L_{in}, L_1)$-list recoverable from a collection, and has rate at least $1-\epsilon/9$, and in which $n_{in}, L_{in}, L_1 = \text{poly}(1/\epsilon)$ are constants.

This gives $n_{out} = n_1 \cdot n_2$, and the correctness follows as concatenation of list-recoverable codes gives a list recoverable codes. Specifically: Given a collection of $n_{out} = n_1 \cdot n_2$ sets (indexed by $(i_1, i_2) \in [n_1] \times [n_2]$), $T_{(i_1,i_2)} \subseteq \{0,1\}^{in}$ of size L_{in}, we need to list recover a list of size at most L, containing all $m \in \{0,1\}^k$ such that

$$\Pr_{(i_1,i_2) \leftarrow [n_1]\times[n_2]}[\text{Enc}_{C_{out}}(m)_{(i_1,i_2)} \in T] \geq 1-\gamma^2.$$

By averaging, for every such m, we have that for a $1-\gamma$ fraction of $i_1 \in [n_1]$,

$$\Pr_{i_2 \leftarrow [n_2]}[\text{Enc}_{C_2}(\text{Enc}_{C_1}(m)_{i_1}) \in T] \geq 1-\gamma.$$

and so performing two steps of list-recovering indeed recovers the original message.

The outer code C_1 can be taken to be a Reed-Solomon code, and by [20, 5], we get these parameters if $\epsilon \leq O(\gamma^2)$ for $L_2 = \text{poly}(1/\epsilon) \cdot L_1$. We now turn our attention to the inner code C_2. We will use the probabilistic method to show the existence of a good code, and such code can be later found by exhaustive search.

▶ **Claim 52.** *There exists a constant $c > 1$, such that for every sufficiently small constants $\epsilon > 0$ and $\gamma > 0$ such that $\epsilon \leq \gamma^c$ and every constant L_{in}, there exist constants $L_1 = L_{in} \cdot poly(1/\epsilon)$ and $n_{in} \geq \frac{\log L_{in}}{\gamma}$, such that for every sufficiently large k_2, there is a code $C_2 : \{0,1\}^{k_2} \to (\{0,1\}^{n_{in}})^{n_2}$ that is $(1-\gamma^2, L_{in}, L_1)$-list recoverable from a collection and has rate $1-\epsilon/9$.*

Proof. We consider a uniformly chosen C_2. For every subset $S \subseteq \{0,1\}^{k_2}$ of size L_1+1, and every collection T of sets $T_1, \ldots, T_2 \subseteq \{0,1\}^{n_{in}}$ of size L_{in} let $B^{S,T}$ be the event that for every $x \in S$, for a $1-\gamma^2$ fraction of $i \in [n_2]$, $\text{Enc}_{C_2}(x)_i \in T_i$. Our goal is to do a union bound over all of these events. We will choose n_{in} to be sufficiently large so that $L_{in} \leq 2^{\gamma n_{in}}$. Let $N_{in} = 2^{n_{in}}$ and let $\alpha = L_{in}/N_{in}$ so that $\log(1/\alpha) = (1-\gamma)\cdot n_{in}$. Note that for fixed x and a collection T, we can use a Chernoff bound[13], to show that the probability that a $1-\gamma^2$ fraction of $i \in [n_2]$, $C_2(x)_i \in T_i$, is at most

$$2^{-(1-\gamma^2)\cdot n_2 \cdot \log \frac{1-\gamma^2}{e\cdot\alpha}} \leq 2^{-(1-\gamma^2)\cdot n_2\cdot \log\frac{10}{\alpha}}$$

where the last inequality follows for sufficiently small γ. It follows that for every S, T:

$$\Pr[B^{S,T}] \leq 2^{-(L+1)\cdot(1-\gamma^2)\cdot n_2 \cdot \log\frac{10}{\alpha}}.$$

The number of choices for S, T is bounded by:

$$\binom{2^{k_2}}{L_1+1} \cdot \binom{N_{in}}{L_{in}}^{n_2} \leq 2^{(L_1+1)\cdot k_2} \cdot \left(\frac{e\cdot N_{in}}{L_{in}}\right)^{n_2\cdot L_{in}} \leq 2^{(L_1+1)\cdot k_2} \cdot 2^{n_2 \cdot L_{in}\cdot \log\frac{e}{\alpha}}.$$

Thus, we can do a union bound if:

$$k_2 < (1-\gamma)\cdot(1-\gamma^2)\cdot n_2 \cdot \log\frac{10}{\alpha} = (1-\gamma^2)\cdot(1-\gamma)^2\cdot n_2\cdot n_{in},$$

and also,

$$L_{in}\cdot\log\frac{e}{\alpha} < \gamma\cdot(L+1)\cdot(1-\gamma^2)\cdot\log\frac{10}{\alpha}.$$

The first inequality follows because we are allowed to choose $k_2 = (1-\epsilon/9)\cdot n_2 \cdot n_{in}$, and ϵ is was chosen to be sufficiently smaller than γ. The second inequality follows as we are allowed to choose $L_1 = L_{in}\cdot \mathrm{poly}(1/\epsilon)$, and $\gamma \geq \epsilon$. ◀

The inner code C_2 is over an alphabet of logarithmic size in k_{out}, and can be found (and decoded) by brute force search in time polynomial in k_{out}.

Acknowledgement. We are grateful to Swastik Kopparty for pointing us to the Algebraic Geometric codes of Garcia and Stichtenoth, and in particular for pointing us to their description in [18].

References

1 Miklós Ajtai. Σ^1_1-formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–48, 1983.

2 M. Braverman. Polylogarithmic independence fools ac^0 circuits. *J. ACM*, 57(5), 2010. doi:10.1145/1754399.1754401.

3 A. Garcia and H. Stichtenoth. On the asymptotic behavior of some towers of function fields over finite fields. *Journal of Number Theory*, 61(2):248–273, 1996.

4 Oded Goldreich. A sample of samplers – a computational perspective on sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, 4(20), 1997.

5 V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

6 Venkatesan Guruswami and Adam D. Smith. Codes for computationally simple channels: Explicit constructions with optimal rate. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 723–732, 2010. doi:10.1109/FOCS.2010.74.

[13] The Chernoff bound we use is that if $X_1, \ldots, X_n$ are independent indicator random variables, and the expectation of their sum X is μn, then for $v > 10$, $\Pr[X \geq v\cdot\mu\cdot n] \leq 2^{-v\cdot\mu\cdot n\cdot\ln(v/e)}$.

7 R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 356–364, 1994. `doi:10.1145/195058.195190`.

8 R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC*, pages 220–229, 1997.

9 E. Kaplan, M. Naor, and O. Reingold. Derandomized constructions of k-wise (almost) independent permutations. *Algorithmica*, 55(1):113–133, 2009. `doi:10.1007/s00453-008-9267-y`.

10 Michael Langberg. Private codes or succinct random codes that are (almost) perfect. In *45th Symposium on Foundations of Computer Science (FOCS 2004)*, pages 325–334, 2004. `doi:10.1109/FOCS.2004.51`.

11 Richard J. Lipton. A new approach to information theory. In *11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 699–708, 1994. `doi:10.1007/3-540-57785-8_183`.

12 Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson. Optimal error correction for computationally bounded noise. *IEEE Trans. Information Theory*, 56(11):5673–5680, 2010. `doi:10.1109/TIT.2010.2070370`.

13 N. Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

14 N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

15 N. Nisan and A. Wigderson. Hardness vs. randomness. *JCSS: Journal of Computer and System Sciences*, 49, 1994.

16 N. Nisan and D. Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

17 Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995. `doi:10.1137/S089548019223872X`.

18 Amir Shpilka. Constructions of low-degree and error-correcting epsilon-biased generators. *Computational Complexity*, 18(4):495–525, 2009. `doi:10.1007/s00037-009-0281-5`.

19 Adam D. Smith. Scrambling adversarial errors using few random bits, optimal information reconciliation, and better private codes. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 395–404, 2007. URL: `http://dl.acm.org/citation.cfm?id=1283383.1283425`.

20 M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13, 1997.

21 Avishay Tal. Tight bounds on the fourier spectrum of ac^0. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:174, 2014. URL: `http://eccc.hpi-web.de/report/2014/174`.

22 Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of AC0. In *Proceedings of the 28th Conference on Computational Complexity, CCC*, pages 242–247, 2013. `doi:10.1109/CCC.2013.32`.

23 Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. Cryptology*, 17(1):43–77, 2004. `doi:10.1007/s00145-003-0237-x`.

Counting Hypergraph Matchings up to Uniqueness Threshold[*]

Renjie Song[1], Yitong Yin[2], and Jinman Zhao[3]

1 **Department of Computer Science and Technology, Nanjing University, China**
song.renjie@foxmail.com
2 **State Key Lab for Novel Software Technology, Nanjing University, China**
yinyt@nju.edu.cn
3 **Department of Computer Science, University of Wisconsin-Madison, USA**
jinman.zhao@gmail.com

Abstract

We study the problem of approximately counting matchings in hypergraphs of bounded maximum degree and maximum size of hyperedges. With an activity parameter λ, each matching M is assigned a weight $\lambda^{|M|}$. The counting problem is formulated as computing a partition function that gives the sum of the weights of all matchings in a hypergraph. This problem unifies two extensively studied statistical physics models in approximate counting: the hardcore model (graph independent sets) and the monomer-dimer model (graph matchings).

For this model, the critical activity $\lambda_c = \frac{d^d}{k(d-1)^{d+1}}$ is the threshold for the uniqueness of Gibbs measures on the infinite $(d+1)$-uniform $(k+1)$-regular hypertree. Consider hypergraphs of maximum degree at most $k+1$ and maximum size of hyperedges at most $d+1$. We show that when $\lambda < \lambda_c$, there is an FPTAS for computing the partition function; and when $\lambda = \lambda_c$, there is a PTAS for computing the log-partition function. These algorithms are based on the decay of correlation (strong spatial mixing) property of Gibbs distributions. When $\lambda > 2\lambda_c$, there is no PRAS for the partition function or the log-partition function unless NP=RP.

Towards obtaining a sharp transition of computational complexity of approximate counting, we study the local convergence from a sequence of finite hypergraphs to the infinite lattice with specified symmetry. We show a surprising connection between the local convergence and the reversibility of a natural random walk. This leads us to a barrier for the hardness result: The non-uniqueness of infinite Gibbs measure is not realizable by any finite gadgets.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases approximate counting; phase transition; spatial mixing

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.46

1 Introduction

Counting problems have long been studied in the context of statistical physics models. Perhaps the two most well studied statistical physics models for approximate counting are the *hardcore model* and the *monomer-dimer model*.

In the hardcore model, given a graph $G = (V, E)$ and a vertex-activity λ, the model assigns each independent set I of G a weight $w_\lambda^{\mathsf{IS}}(I) = \lambda^{|I|}$. A natural probability distribution, the Gibbs distribution, is defined over all independent sets of G as $\mu_\lambda^{\mathsf{IS}}(I) = w_\lambda^{\mathsf{IS}}(I)/Z_\lambda^{\mathsf{IS}}(G)$ where

[*] This research is supported by NSFC grants 61272081 and 61321491. This work was done in part while YY was visiting the Simons Institute for the Theory of Computing.

the normalizing factor $Z_\lambda^{\mathsf{IS}}(G) = \sum_I w_\lambda^{\mathsf{IS}}(I)$ is the *partition function*. In the monomer-dimer model, given a graph $G = (V, E)$ and an edge-activity λ, the model assigns each matching M of G a weight $w_\lambda^{\mathsf{M}}(M) = \lambda^{|M|}$. The Gibbs distribution over all matchings of G is defined accordingly. And the partition function now becomes $Z_\lambda^{\mathsf{M}}(G) = \sum_M w_\lambda^{\mathsf{M}}(M)$. The counting problems are then formulated as computing the partition functions $Z_\lambda^{\mathsf{IS}}(G)$ and $Z_\lambda^{\mathsf{M}}(G)$, or the *log-partition functions* $\log Z_\lambda^{\mathsf{IS}}(G)$ and $\log Z_\lambda^{\mathsf{M}}(G)$.

It was well known that the hardcore model exhibits the following phase transition. For the infinite $(d+1)$-regular tree $\mathbb{T}_d$, there is a critical activity $\lambda_c(\mathbb{T}_d) = d^d/(d-1)^{d+1}$, called the *uniqueness threshold*, such that when $\lambda < \lambda_c$ the correlation between the marginal distribution at the root and any boundary condition on leaves at level t decays exponentially in the depth t, but when $\lambda > \lambda_c$ the boundary-to-root correlation remains substantial even as $t \to \infty$. This property of correlation decay is also called spatial mixing, and was known to be equivalent to the uniqueness of the infinite-volume Gibbs measure on the infinite $(d+1)$-regular tree $\mathbb{T}_d$ [33]. In a seminal work [34], Weitz showed that for all $\lambda < \lambda_c(\mathbb{T}_d)$ the decay of correlation holds for the hardcore model on all graphs of maximum degree bounded by $d+1$ and there is a deterministic FPTAS for approximately computing the partition function on all such graphs. Here the specific notion of decay of correlation established is the strong spatial mixing. The connection of approximability of partition function to the phase transition of the model is further strengthened in a series of works [30, 31, 7, 9] which show that unless NP=RP there is no PRAS for the partition function or the log-partition function of the hardcore model when $\lambda > \lambda_c(\mathbb{T}_d)$ on graphs with maximum degree bounded by $d+1$.

For the monomer-dimer model, it was well known that the model has no such phase transition [13, 14]. And analogously there is an FPRAS due to Jerrum and Sinclair [16] for the partition function of the monomer-dimer model on all graphs. In [1] strong spatial mixing with an exponential rate was established for the model on all graphs with maximum degree bounded by an arbitrary constant and a deterministic FPTAS was also given for the partition function on all such graphs.

In this paper, we study *hypergraph matchings*, a model that unifies both the hardcore model and the monomer-dimer model. A hypergraph $\mathcal{H} = (V, E)$ consists of a vertex set V and a collection E of vertex subsets, called the (hyper)edges. A *matching* of $\mathcal{H}$ is a set $M \subseteq E$ of disjoint hyperedges in $\mathcal{H}$. Given a hypergraph $\mathcal{H}$ and an *activity parameter* $\lambda > 0$, a *configuration* is a matching M of $\mathcal{H}$, and is assigned a weight $w_\lambda(M) = \lambda^{|M|}$. The *Gibbs measure* over all matchings of $\mathcal{H}$ is defined as $\mu(M) = w_\lambda(M)/Z_\lambda(\mathcal{H})$, where the normalizing factor $Z_\lambda(\mathcal{H})$ is the *partition function* for the model, defined as:

$$Z_\lambda(\mathcal{H}) = \sum_{M\text{: matching of }\mathcal{H}} \lambda^{|M|}.$$

This model represents an interesting subclass of Boolean CSP defined by the matching (packing) constraints. It also unifies the hardcore model and the monomer-dimer model. Consider the family of hypergraphs of maximum edge size $d+1$ and maximum degree $k+1$:

- When $d = 1$, the model becomes the monomer-dimer model on graphs of maximum degree $k+1$.
- When $k = 1$, the partition function takes sum over independent sets in the dual graph, and the model becomes the hardcore model on graphs of maximum degree $d+1$.

For hypergraphs, the study of approximate counting hypergraph matchings was initiated in [17]. In [5], an FPTAS was obtained for counting matchings in 3-uniform hypergraphs of maximum degree at most 3 by considering the correlation decay for the independent sets in claw-free graphs. In [22], an FPTAS was given for 3-uniform hypergraphs of maximum degree at most 4 by the correlation decay of the original CSP. All these results assumed $\lambda = 1$, i.e. the problem of counting the number of matchings in a hypergraph.

Our results

We show that for hypergraph matchings $\lambda_c = \lambda_c(\mathbb{T}_{d,k}) = \frac{d^d}{k(d-1)^{d+1}}$ is the uniqueness threshold on the infinite $(d+1)$-uniform $(k+1)$-regular hypertree $\mathbb{T}_{d,k}$.

▶ **Proposition 1.** *There is a unique Gibbs measure on matchings of $\mathbb{T}_{d,k}$ if and only if $\lambda \leq \lambda_c$.*

This fact was implicit in the literature. Here we give a formal proof. It subsumes the well-known uniqueness threshold $\lambda_c(\mathbb{T}_{d,1}) = \frac{d^d}{(d-1)^{d+1}}$ for the hardcore model on the infinite $(d+1)$-regular tree and also the lack of phase-transition for the monomer-dimer model.

We then establish the decay of correlation for hypergraph matchings on all hypergraphs with bounded maximum size of hyperedges and bounded maximum degree when the activity λ is in the uniqueness regime for the uniform regular hypertree. The specific notion of decay of correlations that we establish here is the strong spatial mixing [34] (see Section 2 for a formal definition). Consequently, we give an FPTAS for the partition function when λ is in the interior of the uniqueness regime, and a PTAS for the log-partition function when λ is at the critical threshold.

▶ **Theorem 2.** *For every finite integers $d, k \geq 1$, the following holds for matchings with activity λ on all hypergraphs of maximum edge-size at most $d+1$ and maximum degree at most $k+1$:*

- *if $\lambda < \lambda_c$, the model exhibits strong spatial mixing at an exponential rate and there exists an FPTAS for computing the partition function;*
- *if $\lambda = \lambda_c$, the model exhibits strong spatial mixing at a polynomial rate and there is a PTAS for computing the log-partition function.*

▶ Remark 3. The theorem unifies the strong spatial mixing and FPTAS for the hardcore model [34] and the monomer-dimer model [1], and also covers as special cases the results for approximate counting non-weighted hypergraph matchings in [17, 5, 22].

For hypergraph matchings, the case of critical threshold is of significance. There is a natural combinatorial problem that corresponds to the threshold case: counting matchings in 3-uniform hypergraphs of maximum degree at most 5. Here $d = 2$, $k = 4$, and the critical $\lambda_c = \frac{d^d}{k(d-1)^{d+1}} = 1$, which corresponds to counting the number of hypergraph matchings without weight.

Unlike most recent correlation-decay-based algorithms, where the strong spatial mixings were established by a potential analysis, we do not use the potential method to analyze the decay of correlation. Instead, we prove the following stronger extremal statement.

▶ **Proposition 4.** *For hypergraph matchings, the worst case of (weak or strong) spatial mixing, in terms of decay rate, among all hypergraphs of maximum edge-size at most $d+1$ and maximum degree at most $k+1$, is represented by the weak spatial mixing on $\mathbb{T}_{d,k}$.*

We construct a hypergraph version of Weitz's self-avoiding walk tree. Then we show that weak spatial mixing on the uniform regular hypertree implies strong spatial mixing on all smaller hypertrees by a step-by-step comparison of correlation decay. This was the original approach used by Weitz for the hardcore model [34]. Compared to the more recent potential method [20, 19, 27, 28, 29, 22], this method of analyzing the decay of correlation has the advantage in dealing with the critical case.

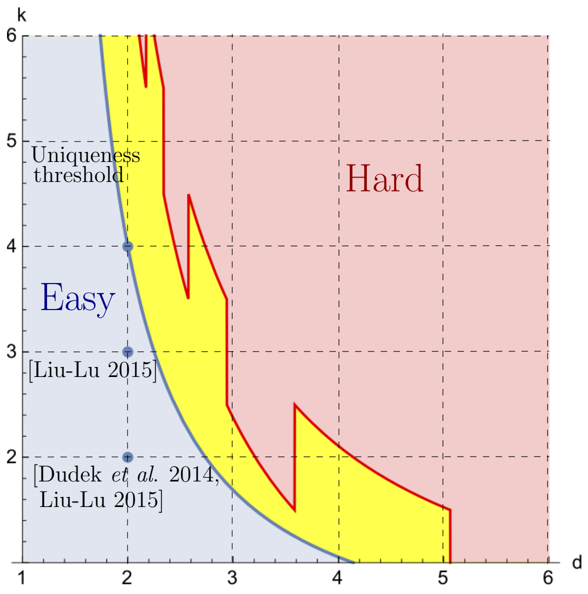


Figure 1 The classification of computational complexity of approximately counting matchings in hypergraphs of max-degree $(k+1)$ and max-edge-size $(d+1)$ when $\lambda = 1$. The blue curve is the uniqueness threshold. The non-continuity of the red curve is due to rounding.

On the other hand, due to a simple reduction from the inapproximability of the hardcore model in the non-uniqueness regime [31], we have the following hardness result.

▶ **Theorem 5.** *If $\lambda > \frac{2k+1+(-1)^k}{k+1}\lambda_c \approx 2\lambda_c$, then there is no PRAS for the partition function or the log-partition function for the family of hypergraphs stated in Theorem 2, unless NP=RP.*

Figure 1 illustrates the classification of approximability of counting hypergraph matchings when $\lambda = 1$. Each integral point (d, k) corresponds to the problem of approximately counting matchings in hypergraphs of max-degree $(k+1)$ and max-edge-size $(d+1)$. The landscape will continuously change when λ changes.

It is worth noticing that in our reduction the hard instances contain many small cycles, while from the algorithmic side the worst cases for the decay of correlation are trees. This obvious inconsistency between upper and lower bounds and the *ad hoc* nature of the simple reduction seem to suggest that the current hardness threshold is not optimal.

We then explore the possibility of bringing the current hardness threshold from $\approx 2\lambda_c$ down to the phase-transition threshold λ_c. We discover a reason why getting the exact transition of approximability could be so challenging for this model on hypergraphs.

To state our discovery, let us first review the current approach for establishing computational phase transition for approximate counting [6, 26, 30, 31, 7, 9, 10], which consists of two main steps:

- (from all infinite measures to finitely many infinite measures) The uniqueness threshold $\lambda_c(\mathbb{T}_d)$ for the Gibbs measure on the infinite regular tree $\mathbb{T}_d$ is achieved by a sub-family of Gibbs measures with simple structure: the Gibbs measures that are invariant under a group $\mathbb{G}$ of automorphisms on $\mathbb{T}_d$. For the hardcore model, these are the so-called *semi-translation invariant* Gibbs measures, which are invariant under parity-preserving automorphisms on $\mathbb{T}_d$, and the threshold $\lambda_c(\mathbb{T}_d)$ for the uniqueness of all Gibbs measures on $\mathbb{T}_d$ is the same as the threshold $\lambda_c(\mathbb{T}_d^{\mathbb{G}})$ for the uniqueness of only those Gibbs measures that are invariant under the group $\mathbb{G}$ of parity-preserving automorphisms.

- (from finitely many infinite measures to finite measures) A sequence of (possibly random) finite graphs G_n is constructed to converge locally to $\mathbb{T}_d^{\mathbb{G}}$, the infinite tree $\mathbb{T}_d$ equipped with the symmetry specified by group $\mathbb{G}$. For the hardcore model, and more generally antiferromagnetic spin systems, G_n are the random regular bipartite graphs [6, 26, 30, 31, 7, 9, 10], which converge locally to the infinite tree $\mathbb{T}_d$ respecting the symmetry between vertices of the same parity. The "random" and "regular" parts in this construction guarantee to preserve the local tree structure in distribution, while the bipartiteness respects the parity of vertices.

For the model of hypergraph matchings, the first step follows. We show that there indeed is a group $\widehat{\mathbb{G}}$ of automorphisms on the infinite $(d+1)$-uniform $(k+1)$-regular hypertree $\mathbb{T}_{d,k}$ such that $\lambda_c(\mathbb{T}_{d,k}) = \lambda_c(\mathbb{T}_{d,k}^{\widehat{\mathbb{G}}})$, i.e. the uniqueness of Gibbs measure on $\mathbb{T}_{d,k}$ is represented precisely by the uniqueness of only those Gibbs measures invariant under $\widehat{\mathbb{G}}$. This gives a natural generalization of semi-translation Gibbs measures to the hypergraph model.

However, we show that there does not exist *any* sequence of (deterministic or random) finite hypergraphs that converge locally to $\mathbb{T}_{d,k}^{\widehat{\mathbb{G}}}$ unless $k = 1$ where the model degenerates to the hardcore model on graphs. In fact, we give a complete characterization of the symmetry described by a group $\mathbb{G}$ of automorphisms on $\mathbb{T}_{d,k}$ that there exists a sequence of finite hypergraphs that converge locally to $\mathbb{T}_{d,k}^{\mathbb{G}}$.

▶ **Theorem 6.** *Let $\mathbb{G}$ be a group of automorphisms on $\mathbb{T}_{d,k}$ with finitely many orbits. There exist a sequence of random finite hypergraphs $\mathcal{H}_n$ that converge locally to $\mathbb{T}_{d,k}^{\mathbb{G}}$ if and only if the uniform random walk on $\mathbb{T}_{d,k}$ projected onto the orbits of $\mathbb{G}$ is reversible.*

See Theorem 27 and its proof for more details of Theorem 6.

Discussion

To summarize our discoveries for the model of hypergraph matchings:

- Theorem 2 implicitly but rigorously shows that the worst case for the decay of correlation among a family of hypergraphs with bounded maximum degree and bounded maximum edge-size, is achieved by the infinite uniform regular hypertree.
- However, in the current inapproximability stated by Theorem 5, the hard instances are not locally tree-like, but rather, the gadgets locally converge to an infinite hypergraph which is not a hypertree (see Section 6).
- And finally, Theorem 6 gives an explanation of this inconsistence between upper and lower bounds: the extremal case for the decay of correlation in Theorem 2, which is achieved by an infinite-hypertree measure, can never be realized by *any* finite hypergraphs.[1]

Altogether, these discoveries deliver the following very interesting message: In order to establish a sharp connection between computational complexity of approximate counting and phase transitions for hypergraph matchings or other more general models, a more fine-grained definition of uniqueness on finite graphs is necessary.

[1] In fact, aided by numerical simulations, so far we have not encountered any family of measures on the infinite uniform regular hypertree $\mathbb{T}_{d,k}$ realizable by finite hypergraphs, whose uniqueness threshold is below $2\lambda_c$. This seems to provide some empirical evidence for that on finite hypergraphs, the worst case for uniqueness might not be locally tree-like.

Remark on exposition

For convenience of visualizing the results, all our results in the rest of the paper are presented for *independent sets* in the dual hypergraphs. Note that matchings are equivalent to independent sets under hypergraph duality. The only effect of duality on a family of hypergraphs with bounded maximum edge size and bounded maximum degree is to switch the bounds on the edge size and the degree. We emphasize that our notion of hypergraph independent set is different from the more popular definition used in [2, 3]. We call a vertex subset $I \subseteq V$ in a hypergraph $\mathcal{H} = (V, E)$ an independent set if no two vertices in I are contained in the same hyperedge, while in [2, 3], an $I \subseteq V$ is an independent set if it does not contain any hyperedge as subset.

Related works

Approximate counting of hypergraph matchings was studied in [17] for hypergraphs with restrictive structures, and in [22, 5] for hypergraphs with bounded edge size and maximum degree. In [3, 24], approximate counting of a variant of hypergraph independent sets was studied, where the definition of hypergraph independent set is different from ours. In a very recent breakthrough [2], FPTAS for this problem is obtained when there is no strong spatial mixing. In [8], the hardness is established for a class of hypergraph models including ours.

The spatial mixing (decay of correlation) is already a widely studied topic in Computer Science, because it may support FPTAS for #P-hard counting problems. The decay of correlation was established via the self-avoiding walk tree for the hardcore model [34, 29], monomer-dimer model [1, 28], and two-spin systems [20, 19, 28]. Similar tree-structured recursions were employed to prove the decay of correlation for multi-spin systems [11, 25, 12] and more general CSPs [21, 23, 22].

2 Preliminaries

For a *hypergraph* $\mathcal{H} = (V, E)$, the *size* of a hyperedge $e \in E$ is its cardinality $|e|$, and the *degree* of a vertex $v \in V$, denoted by $\deg v = \deg_{\mathcal{H}}(v)$, is the number of hyperedges $e \in E$ *incident to* v, i.e. satisfying $v \in e$. A hypergraph $\mathcal{H}$ is *k-uniform* if all hyperedges are of the same size k, and is *d-regular* if all vertices have the same degree d. The *incidence graph* of a hypergraph $\mathcal{H} = (V, E)$ is a bipartite graph with V and E as vertex sets on the two sides, such that each $(v, e) \in V \times E$ is a bipartite edge if and only if v is incident to e.

A *matching* of hypergraph $\mathcal{H} = (V, E)$ is a set $M \subseteq E$ of disjoint hyperedges in $\mathcal{H}$. Given an *activity parameter* $\lambda > 0$, the *Gibbs measure* is a probability distribution over matchings of $\mathcal{H}$ proportional to the weight $w_\lambda^{\mathsf{M}}(M) = \lambda^{|M|}$, defined as $\mu_\lambda^{\mathsf{M}}(M) = w_\lambda^{\mathsf{M}}(M)/Z_\lambda^{\mathsf{M}}(\mathcal{H})$, where the normalizing factor $Z_\lambda^{\mathsf{M}}(\mathcal{H}) = \sum_M w_\lambda^{\mathsf{M}}(M)$ is the partition function.

Similarly, an *independent set* of hypergraph $\mathcal{H} = (V, E)$ is a set $I \subseteq V$ of vertices satisfying $|I \cap e| \le 1$ for all hyperedges e in $\mathcal{H}$. The *Gibbs measure* over independent sets of $\mathcal{H}$ with activity $\lambda > 0$ is given by

$$\mu_\lambda^{\mathsf{IS}}(I) = \frac{w_\lambda^{\mathsf{IS}}(I)}{Z_\lambda^{\mathsf{IS}}(\mathcal{H})} = \frac{\lambda^{|I|}}{Z_\lambda^{\mathsf{IS}}(\mathcal{H})}, \tag{1}$$

where the normalizing factor $Z_\lambda^{\mathsf{IS}}(\mathcal{H}) = \sum_I w_\lambda^{\mathsf{IS}}(I)$ is the partition function for independent sets of $\mathcal{H}$ with activity λ.

Independent sets and matchings are equivalent under hypergraph duality. The *dual* of a hypergraph $\mathcal{H} = (V, E)$, denoted by $\mathcal{H}^* = (E^*, V^*)$, is the hypergraph whose vertex set is denoted by E^* and edge set is denoted by V^*, such that every vertex $v \in V$ (and every hyperedge $e \in E$) in $\mathcal{H}$ is one-to-one corresponding to a hyperedge $v^* \in V^*$ (and a vertex $e^* \in E^*$), such that $e^* \in v^*$ if and only if $v \in e$. Note that under duality, matchings and hypergraphs are the same CSP and hence result in the same Gibbs measure, which remains to be true even with activity λ. Also a family of hypergraphs of bounded maximum edge size and bounded maximum degree is transformed under duality to a family of hypergraphs with the bounds on the edge size and degree exchanged.

▶ Remark 7. With the above equivalence under duality, from now on we state all our results in terms of the independent sets in the dual hypergraph and omit the superscript $\cdot^{\mathsf{IS}}$ in notations.

Given the Gibbs measure over independent sets of hypergraph $\mathcal{H}$ and a vertex v, we define the *marginal probability* p_v as

$$p_v = p_{\mathcal{H},v} = \Pr[v \in I]$$

which is the probability that v is in an independent set I sampled from the Gibbs measure (such a vertex is also said to be *occupied*). Given a vertex set $\Lambda \subset V$, a *configuration* is a $\sigma_\Lambda \in \{0,1\}^\Lambda$ which corresponds to an independent set I_Λ partially specified over Λ such that $\sigma_\Lambda(v)$ indicates whether a $v \in \Lambda$ is occupied by the independent set. We further define the marginal probability $p^{\sigma_\Lambda}_{\mathcal{H},v}$ as

$$p^{\sigma_\Lambda}_v = p^{\sigma_\Lambda}_{\mathcal{H},v} = \Pr[v \in I \mid I_\Lambda = \sigma_\Lambda]$$

which is the probability that v is occupied under the Gibbs measure conditioning on the configuration of vertices in $\Lambda \subset V$ being fixed as σ_Λ.

▶ **Definition 8.** The independent sets of a finite hypergraph $\mathcal{H} = (V, E)$ with activity $\lambda > 0$ exhibit *weak spatial mixing (WSM)* with rate $\delta : \mathbb{N} \to \mathbb{R}^+$ if for any $v \in V$, $\Lambda \subseteq V$, and any two configurations $\sigma_\Lambda, \tau_\Lambda \in \{0,1\}^\Lambda$ which correspond to two independent sets partially specified on Λ,

$$|p^{\sigma_\Lambda}_v - p^{\tau_\Lambda}_v| \le \delta(\mathrm{dist}_{\mathcal{H}}(v, \Lambda)),$$

where $\mathrm{dist}_{\mathcal{H}}(v, \Lambda)$ is the shortest distance between v and any vertex in Λ in hypergraph $\mathcal{H}$.

▶ **Definition 9.** The independent sets of a finite hypergraph $\mathcal{H} = (V, E)$ with activity $\lambda > 0$ exhibit *strong spatial mixing (SSM)* with rate $\delta : \mathbb{N} \to \mathbb{R}^+$ if for any $v \in V$, $\Lambda \subseteq V$, and any two configurations $\sigma_\Lambda, \tau_\Lambda \in \{0,1\}^\Lambda$ which correspond to two independent sets partially specified on Λ,

$$|p^{\sigma_\Lambda}_v - p^{\tau_\Lambda}_v| \le \delta(\mathrm{dist}_{\mathcal{H}}(v, \Delta)),$$

where $\Delta \subseteq \Lambda$ stands for the subset on which σ_Λ and τ_Λ differ and $\mathrm{dist}_{\mathcal{H}}(v, \Delta)$ is the shortest distance between v and any vertex in Δ in hypergraph $\mathcal{H}$.

The definitions of WSM and SSM extend to infinite hypergraphs with the same conditions to be satisfied for every finite region $\Psi \subset V$ conditioning on the vertices in $\partial\Psi$ being unoccupied.

3 Gibbs measures on the infinite tree

We follow Remark 7 and state our discoveries in terms of independent sets in the dual hypergraphs. Let $\mathbb{T}_{k,d}$ be the infinite $(k+1)$-uniform $(d+1)$-regular hypertree, whose incidence graph is the infinite tree in which all vertices with parity 0 are of degree $(k+1)$ and all vertices with parity 1 are of degree $(d+1)$. A probability measure μ on hypergraph independent sets of $\mathbb{T}_{k,d}$ is *Gibbs* if for any finite sub-hypertree $\mathcal{T}$, conditioning μ upon the event that all vertices on the outer boundary of $\mathcal{T}$ are unoccupied gives the same distribution on independent sets of $\mathcal{T}$ as defined by (1) with $\mathcal{H} = \mathcal{T}$. We further consider the *simple* Gibbs measures satisfying *conditional independence*: Conditioning μ on a configuration of a subset Λ of vertices results in a measure in which the configurations on the components separated by Λ are independent of each other. The Gibbs distribution on a finite hypergraph is always simple. A Gibbs measure on $\mathbb{T}_{k,d}$ is *translation-invariant* if it is invariant under all automorphisms of $\mathbb{T}_{k,d}$. Fix an automorphism group $\mathbb{G}$ of $\mathbb{T}_{k,d}$. A $\mathbb{G}$*-translation-invariant* Gibbs measure on $\mathbb{T}_{k,d}$ is a measure that is invariant under all automorphisms from $\mathbb{G}$. For example, the *semi-translation-invariant* Gibbs measures on regular tree are invariant under all parity-preserving automorphisms on $\mathbb{T}_{1,d}$. The natural group actions of $\mathbb{G}$ respectively on vertices and hyperedges partition the sets of vertices and hyperedges into orbits. For example, in the semi-translation-invariant symmetry on regular tree, vertices with the same parity form an orbit. We will show that $\lambda_c(\mathbb{T}_{k,d}) = \frac{d^d}{k(d-1)^{d+1}}$ is the uniqueness threshold for the Gibbs measures on hypergraph independent sets of $\mathbb{T}_{k,d}$. Furthermore, this uniqueness threshold is achieved by a family of Gibbs measures with simple structure.

▶ **Theorem 10.** *There is always a unique simple translation-invariant Gibbs measure on independent sets of $\mathbb{T}_{k,d}$. Let $\lambda_c = \lambda_c(\mathbb{T}_{k,d}) = \frac{d^d}{k(d-1)^{d+1}}$. There is a unique Gibbs measure on $\mathbb{T}_{k,d}$ if and only if $\lambda \le \lambda_c$. Furthermore, there is an automorphism group $\widehat{\mathbb{G}}$ on $\mathbb{T}_{k,d}$ which classifies all vertices of $\mathbb{T}_{k,d}$ into 2 orbits, such that the threshold for the uniqueness of $\widehat{\mathbb{G}}$-translation invariant Gibbs measures on $\mathbb{T}_{k,d}$, denoted as $\lambda_c(\mathbb{T}^{\widehat{\mathbb{G}}}_{k,d})$, is $\lambda_c(\mathbb{T}^{\widehat{\mathbb{G}}}_{k,d}) = \lambda_c(\mathbb{T}_{k,d})$.*

This proves the uniqueness threshold stated in Proposition 1.

3.1 Branching matrices

The automorphism group $\mathbb{G}$ on $\mathbb{T}_{k,d}$ can be described conveniently by a notion of branching matrices. For an automorphism group $\mathbb{G}$ on $\mathbb{T}_{k,d}$, the natural group actions of $\mathbb{G}$ respectively on vertices and hyperedges partition the sets of vertices and hyperedges into orbits. Let τ_v and τ_e be the respective numbers of orbits for vertices and hyperedges. For each $i \in [\tau_v]$, we say a vertex is of *type-i* if it is in the i-th orbit for vertices; and the same also applies to hyperedges. Assuming the symmetry on $\mathbb{T}_{k,d}$ given by automorphism group $\mathbb{G}$, the *hypergraph branching matrices*, or just *branching matrices*, are the following two nonnegative integral matrices:

$$\boldsymbol{D} = \boldsymbol{D}^{\tau_v \times \tau_e} = [d_{ij}] \quad \text{and} \quad \boldsymbol{K} = \boldsymbol{K}^{\tau_e \times \tau_v} = [k_{ji}],$$

which satisfy that for any $i \in [\tau_v]$ and $j \in [\tau_e]$:

- every vertex in $\mathbb{T}_{k,d}$ of type-i is incident to precisely d_{ij} hyperedges of type-j;
- every hyperedge in $\mathbb{T}_{k,d}$ of type-j contains precisely k_{ji} vertices of type-i.

The $\boldsymbol{D}$ and $\boldsymbol{K}$ are transition matrices from vertex-types to hyperedge-types and vice versa in $\mathbb{T}_{k,d}$. The definition can be seen as a hypergraph generalization of the branching matrix for multi-type Galton-Watson tree [27]. Since types (orbits) are invariant under all automorphisms from $\mathbb{G}$, it is clear that the above $\boldsymbol{D}$ and $\boldsymbol{K}$ are well-defined for every automorphism group $\mathbb{G}$ on $\mathbb{T}_{k,d}$ with finitely many orbits.

▶ **Proposition 11.** *Every automorphism group $\mathbb{G}$ on $\mathbb{T}_{k,d}$ with finitely many orbits can be identified by a pair of branching matrices $\boldsymbol{D}$ and $\boldsymbol{K}$ with rules as described above and satisfy: (1) $\sum_j d_{ij} = d+1$ and $\sum_i k_{ji} = k+1$; (2) $d_{ij} = 0$ if and only if $k_{ji} = 0$; and (3) $\boldsymbol{DK}$ and $\boldsymbol{KD}$ are irreducible.*

Conversely, any pair of nonnegative integral matrices $\boldsymbol{D}$ and $\boldsymbol{K}$ satisfying these conditions are branching matrices for some automorphism group $\mathbb{G}$ on $\mathbb{T}_{k,d}$.

Proof. Let $\mathbb{G}$ be an automorphism group on $\mathbb{T}_{k,d}$ with finitely many orbits. It is trivial to see that the branching matrices $\boldsymbol{D}$ and $\boldsymbol{K}$ are well-defined and satisfy $\sum_j d_{ij} = d+1$ and $\sum_i k_{ji} = k+1$.

A vertex v of type-i is incident to a hyperedge e of type-j if and only if e of type-j contains a vertex v of type i, thus $k_{ji} \neq 0$ if and only if $d_{ij} \neq 0$.

The irreducibility of $\boldsymbol{DK}$ and $\boldsymbol{KD}$ follows that of the matrix $\begin{bmatrix} \mathbf{0} & \boldsymbol{D} \\ \boldsymbol{K} & \mathbf{0} \end{bmatrix}$, which is a consequence to the that every type of vertex and hyperedge is accessible from all other types of vertices and hyperedges, which follows the simple fact that the incidence graph $\mathbb{T}_{k,d}$ is strongly connected.

Conversely, let $\boldsymbol{D}$ and $\boldsymbol{K}$ be a pair of nonnegative integral matrices satisfying the conditions above. We can start from any vertex (or hyperedges) o of type-i and construct an infinite hypertree rooted at o with each vertex and hyperedge labeled with the respective type according to the rules specified by the branching matrices $\boldsymbol{D}$ and $\boldsymbol{K}$. Since $d_{ij} = 0$ if and only if $k_{ji} = 0$, the construction is always possible. Since $\sum_j d_{ij} = d+1$ and $\sum_i k_{ji} = k+1$, the resulting infinite hypertree must be k-uniform and d-regular. Since $\boldsymbol{DK}$ and $\boldsymbol{KD}$ are irreducible, no matter how we choose the type for the root o, the resulting hypertree contains all types of vertices and hyperedges.

We can then construct an automorphism group $\mathbb{G}$ on $\mathbb{T}_{k,d}$ according with orbits being the types just specified. For every pair of vertices (or hyperedges) u, v with the same type, by generating the hypertree according to $\boldsymbol{D}$, $\boldsymbol{K}$ starting from u and v respectively, we obtain an automorphism $\phi_{u\to v}$ on $\mathbb{T}_{k,d}$ which maps u to v and preserves the types of all vertices and hyperedges. Let $\mathbb{G} = \langle \{\phi_{u\to v} \mid \forall u, v \text{ with the same type}\} \rangle$ be the group generated from all such automorphisms. Then $\boldsymbol{D}$ and $\boldsymbol{K}$ are branching matrices for automorphism group $\mathbb{G}$ on $\mathbb{T}_{k,d}$. ◀

3.2 Extremal Gibbs measures

Consider a special automorphism group $\widehat{\mathbb{G}}$ on $\mathbb{T}_{k,d}$ defined by the following branching matrices $(\widehat{\boldsymbol{D}}, \widehat{\boldsymbol{K}})$. Assume that there are two vertex-types and two hyperedge-types, both denoted as $\{+,-\}$, and the branching matrices are defined as $\widehat{\boldsymbol{D}} = \begin{bmatrix} 1 & d \\ d & 1 \end{bmatrix}$ and $\widehat{\boldsymbol{K}} = \begin{bmatrix} k & 1 \\ 1 & k \end{bmatrix}$, i.e.:

1. every '$\pm$'-vertex is incident to a '$\pm$'-hyperedge and d '$\mp$'-hyperedges;
2. every '$\pm$'-hyperedge contains k '$\pm$'-vertices and a '$\mp$'-vertex.

See Figure 2 for an illustration.

Fix a '$+$'-vertex v in $\mathbb{T}_{k,d}$ as the root. Let μ^+ (resp. μ^-) be the Gibbs measure on $\mathbb{T}_{k,d}$ defined by conditioning on all vertices to be occupied for the t-th '$+$'-vertices (resp. '$-$'-vertices) along all path from the root and taking the weak limit as $t \to \infty$. Note that for the 2-coloring given by $\widehat{\boldsymbol{D}}$ and $\widehat{\boldsymbol{K}}$, on any path any '$\pm$'-vertex has a '$\mp$'-vertex within 2 steps,

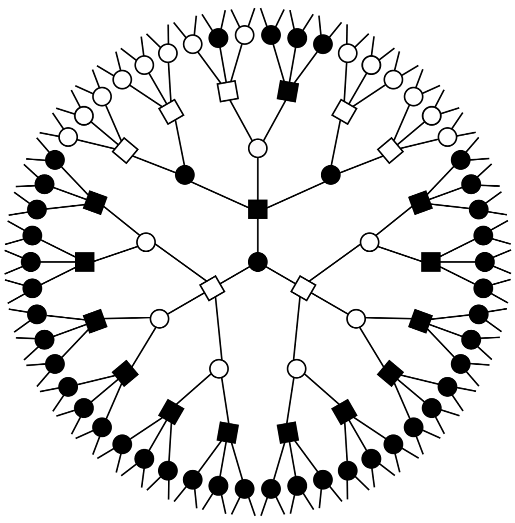


Figure 2 Classifying vertices and hyperedges of $\mathbb{T}_{3,2}$ into two types '+'(black) and '−'(white). The hypergraph is represented as its incidence graph where circles stand for vertices and squares stand for hyperedges.

so the limiting sequence is well-defined. And by symmetry, starting from a root of type-'−' gives the same pair of measures.

The $\mu^{\pm}$ generalize the extremal semi-translation-invariant Gibbs measures on infinite regular trees. For hypertree $\mathbb{T}_{k,d}$ with $k \geq 2$, there are no parity-preserving automorphisms. Nevertheless, the symmetry given by $\widehat{\boldsymbol{D}}$ and $\widehat{\boldsymbol{K}}$ generalizes the parity-preserving automorphisms to hypertrees and has the similar phase-transition as semi-translation-invariant Gibbs measures on trees.

The $\mu^{\pm}$ are simple and are $\widehat{\mathbb{G}}$-translation-invariant for the automorphism group $\widehat{\mathbb{G}}$ with orbits given by $\widehat{\boldsymbol{D}}$ and $\widehat{\boldsymbol{K}}$. In fact, they are extremal $\widehat{\mathbb{G}}$-translation-invariant Gibbs measures on $\mathbb{T}_{k,d}$. We will see that the model has uniqueness if and only if $\mu^+ = \mu^-$.

3.3 Uniqueness of Gibbs measures

▶ **Lemma 12.** *Let μ be a simple Gibbs measure on independent sets of $\mathbb{T}_{k,d}$. Let v be a vertex in $\mathbb{T}_{k,d}$ and v_{ij} the j-th vertex (besides v) in the i-th hyperedge incident to v, for $i = 1, 2, \dots, d+1$ and $j = 1, 2, \dots, k$. Let $p_v = \mu[\,v$ is occupied$]$ and $p_{v_{ij}} = \mu[\,v_{ij}$ is occupied$]$. It holds that*

$$p_v = \lambda(1-p_v)^{-d} \prod_{i=1}^{d+1} \left(1 - p_v - \sum_{j=1}^{k} p_{v_{ij}}\right). \tag{2}$$

Proof. Since μ is a Gibbs measure, for any vertex v in $\mathbb{T}_{k,d}$, it holds that

$$p_v = \mu[\,v \text{ is occupied}\,] = \frac{\lambda}{1+\lambda} \cdot \mu[\,\text{all the neighbors of } v \text{ are unoccupied}\,]$$

On the other hand, since μ is simple, conditioning on the root being unoccupied the sub-hypertrees are independent of each other, thus

$$\begin{aligned}
&\mu[\,\text{all the neighbors of } v \text{ are unoccupied}\,]\\
=&\mu[\,v \text{ is occupied}\,]\cdot\mu[\,\text{all the neighbors of } v \text{ are unoccupied} \mid v \text{ is occupied}\,]\\
&+\mu[\,v \text{ is unoccupied}\,]\prod_{i=1}^{d+1}\mu[\,\forall 1\le j\le k, v_{ij} \text{ is unoccupied} \mid v \text{ is unoccupied}\,]\\
=&p_v+(1-p_v)\prod_{i=1}^{d+1}\left(1-\sum_{j=1}^{k}\mu[\,v_{ij} \text{ is occupied} \mid v \text{ is unoccupied}\,]\right).
\end{aligned}$$

Note that for any two adjacent vertices v, v_{ij}, we have $\mu[\,v_{ij} \text{ is occupied}\,] = \mu[\,v_{ij} \text{ is occupied} \mid v \text{ is unoccupied}\,]\cdot\mu[\,v \text{ is unoccupied}\,]$, thus

$$\mu[\,v_{ij} \text{ is occupied} \mid v \text{ is unoccupied}\,] = \frac{\mu[\,v_{ij} \text{ is occupied}\,]}{1-\mu[\,v \text{ is occupied}\,]} = \frac{p_{v_{ij}}}{1-p_v}.$$

The lemma follows by combining everything together. ◀

Equation (2) gives an infinite system involving all vertices in $\mathbb{T}_{k,d}$. If the simple Gibbs measure μ is $\mathbb{G}$-translation-invariant for some automorphism group $\mathbb{G}$ on $\mathbb{T}_{k,d}$, the marginal probability $p_v = \mu[\,v \text{ is occupied}\,]$ depends only on the type (orbit) of v.

▶ **Corollary 13.** *Let μ be a simple $\mathbb{G}$-translation-invariant Gibbs measure on $\mathbb{T}_{k,d}$ with branching matrices $\boldsymbol{D}^{\tau_v\times\tau_e} = [d_{ij}]$ and $\boldsymbol{K}^{\tau_e\times\tau_v} = [k_{ji}]$. For every $i \in [\tau_v]$, let $p_i = \mu[\,v$ is occupied$]$ for vertex v in $\mathbb{T}_{k,d}$ of type-i. It holds for every $s \in [\tau_v]$ that*

$$p_s = \lambda(1-p_s)^{-d}\prod_{j\in[\tau_e]}\left(1-\sum_{i\in[\tau_v]}k_{ji}\cdot p_i\right)^{d_{ij}}.$$

Applying with the branching matrices $\widehat{\boldsymbol{D}}$ and $\widehat{\boldsymbol{K}}$ defined in Section 3.2, the system in Corollary 13 becomes

$$\begin{cases} p_+ = \lambda(1-p_+)^{-d}(1-k\,p_+-p_-)(1-p_+-k\,p_-)^d,\\ p_- = \lambda(1-p_-)^{-d}(1-k\,p_--p_+)(1-p_--k\,p_+)^d. \end{cases}$$

Let $x = \frac{kp_+}{1-p_--k\,p_+}$ and $y = \frac{kp_-}{1-p_+-k\,p_-}$. The system becomes $\begin{cases} y = f(x)\\ x = f(y)\end{cases}$, where $f(x) = \frac{k\lambda}{(1+x)^d}$ is the hardcore tree-recursion. Since $f(x)$ is positive and decreasing in x, it follows that there is a unique positive $\hat{x}$ such that $\hat{x} = f(\hat{x})$, which means there is always a unique simple translation-invariant Gibbs measure on $\mathbb{T}_{k,d}$. It is well-known (see [9] and [18, 32]) the system has three distinct solutions $(\hat{x},\hat{x}), (x^+,x^-)$ and (x^-,x^+) where $0 < x^- < \hat{x} < x^+$, when $k\lambda > d^d/(d-1)^{d+1}$, i.e. $\lambda > \lambda_c(\mathbb{T}_{k,d}) = \frac{d^d}{k(d-1)^{d+1}}$; and the three solutions collide into a unique solution $(\hat{x},\hat{x})$ when $\lambda \le \lambda_c(\mathbb{T}_{k,d})$, which means there is a unique simple $\widehat{\mathbb{G}}$-translation-invariant Gibbs measure on $\mathbb{T}_{k,d}$ if and only if $\lambda \le \lambda_c(\mathbb{T}_{k,d})$. Recall that $\mu^\pm$ are simple and are extremal $\widehat{\mathbb{G}}$-translation-invariant Gibbs measures, and hence it also holds that $\mu^+ = \mu^-$ if and only if $\lambda \le \lambda_c(\mathbb{T}_{k,d})$, therefore, it holds that $\lambda_c(\mathbb{T}_{k,d}) = \lambda_c(\mathbb{T}^{\widehat{\mathbb{G}}}_{k,d})$. In particular if $\lambda > \lambda_c(\mathbb{T}_{k,d})$, then $\mu^+ \ne \mu^-$ and the Gibbs measure on $\mathbb{T}_{k,d}$ is non-unique.

To complete the proof of Theorem 10, we only need to show the Gibbs measure on $\mathbb{T}_{k,d}$ is unique if $\lambda \le \lambda_c(\mathbb{T}_{k,d})$. This is implied by the weak spatial mixing on $\mathbb{T}_{k,d}$ when $\lambda \le \lambda_c$, proved later in Theorem 21. With the weak spatial mixing on $\mathbb{T}_{k,d}$, the uniqueness of the Gibbs measure is implied by a generic equivalence between weak spatial mixing and uniqueness of Gibbs measure (see e.g. [33]).

4 The hypergraph self-avoiding walk tree

We call a hypergraph a hypertree if its incidence graph has no cycles. Let $\mathcal{T} = (V, E)$ be a rooted hypertree with vertex v as its root. We assume that root v is incident to d distinct hyperedges $e_1, e_2, \ldots, e_d$, such that for $i = 1, 2, \ldots, d$,

- $|e_i| = k_i + 1$; and
- $e_i = \{v, v_{i1}, v_{i2}, \ldots, v_{ik_i}\}$.

For $1 \le i \le d$ and $1 \le j \le k_i$, let $\mathcal{T}_{ij}$ be the sub-hypertree rooted at v_{ij}. Recall that all hypertrees considered by us satisfy the property that any two hyperedges share at most one common vertex, thus all v_{ij} are distinct and the sub-hypertrees $\mathcal{T}_{ij}$ are disjoint.

Let $\Lambda \subset V$. Let $\sigma_\Lambda \in \{0,1\}^\Lambda$ be a configuration indicating an independent set partially specified on vertex set Λ, and for each $1 \le i \le d$ and $1 \le j \le k_i$, let $\sigma_{\Lambda_{ij}}$ be the restriction of σ_Λ on the sub-hypertree $\mathcal{T}_{ij}$. Consider the ratios of marginal probabilities:

$$R_{\mathcal{T}}^{\sigma_\Lambda} = p_{\mathcal{T},v}^{\sigma_\Lambda} \Big/ \left(1 - p_{\mathcal{T},v}^{\sigma_\Lambda}\right) \quad \text{and} \quad R_{\mathcal{T}_{ij}}^{\sigma_{\Lambda_{ij}}} = p_{\mathcal{T}_{ij},v_{ij}}^{\sigma_{\Lambda_{ij}}} \Big/ \left(1 - p_{\mathcal{T}_{ij},v_{ij}}^{\sigma_{\Lambda_{ij}}}\right).$$

The following recursion can be easily verified due to the disjointness between sub-hypertrees:

$$R_{\mathcal{T}}^{\sigma_\Lambda} = \lambda \prod_{i=1}^{d} \frac{1}{1 + \sum_{j=1}^{k_i} R_{\mathcal{T}_{ij}}^{\sigma_{\Lambda_{ij}}}}. \tag{3}$$

This is the "tree recursion" for hypergraph independent sets. The tree recursions for the hardcore model [34] and the monomer-dimer model [1] can both be interpreted as special cases.

For general hypergraphs which are not trees, we construct a hypergraph version of *self-avoiding-walk tree*, which allows computing marginal probabilities in arbitrary hypergraphs with the tree recursion. Moreover, we show that the uniform regular hypertree is the worst case for SSM among all hypergraphs of bounded maximum edge-size and bounded maximum degree.

▶ **Theorem 14.** *For any positive integers k, d and any positive λ, if the independent sets of $\mathbb{T}_{k,d}$ with activity λ exhibit strong spatial mixing with rate $\delta(\cdot)$, then the independent sets of any hypergraph of maximum edge size at most $(k+1)$ and maximum degree at most $(d+1)$, with activity λ, exhibit strong spatial mixing with the same rate $\delta(\cdot)$.*

Under duality, the same holds for the hypergraph matchings.

We then define the hypergraph self-avoiding walk tree. A walk in a hypergraph $\mathcal{H} = (V, E)$ is a sequence $(v_0, e_1, v_1, \ldots, e_\ell, v_\ell)$ of alternating vertices and hyperedges such that every two consecutive vertices v_{i-1}, v_i are incident to the hyperedge e_i between them. A walk $w = (v_0, e_1, v_1, \ldots, e_\ell, v_\ell)$ is called *self-avoiding* if:

- $w = (v_0, e_1, v_1, \ldots, e_\ell, v_\ell)$ forms a simple path in the incidence graph of $\mathcal{H}$; and
- for every $i = 1, 2, \ldots, \ell$, vertex v_i is incident to none of $\{e_1, e_2, \ldots, e_{i-1}\}$.

Note that the second requirement is new to the hypergraphs.

A self-avoiding walk $w = (v_0, e_1, v_1, \ldots, e_\ell, v_\ell)$ can be extended to a *cycle-closing* walk $w' = (v_0, e_1, v_1, \ldots, e_\ell, v_\ell, e', v')$ so that the suffix $(v_i, e_{i+1}, v_{i+1}, \ldots, e_\ell, v_\ell, e', v')$, for some $0 \le i \le \ell - 1$, of the walk forms a simple cycle in the incidence graph of $\mathcal{H}$. We call v' the cycle-closing vertex.

Given a hypergraph $\mathcal{H} = (V, E)$, an ordering of incident hyperedges at every vertex can be arbitrarily fixed, so that for any two hyperedges e_1, e_2 incident to a vertex u we use $e_1 <_u e_2$ to denote that e_1 is ranked higher than e_2 according to the ordering of hyperedges incident

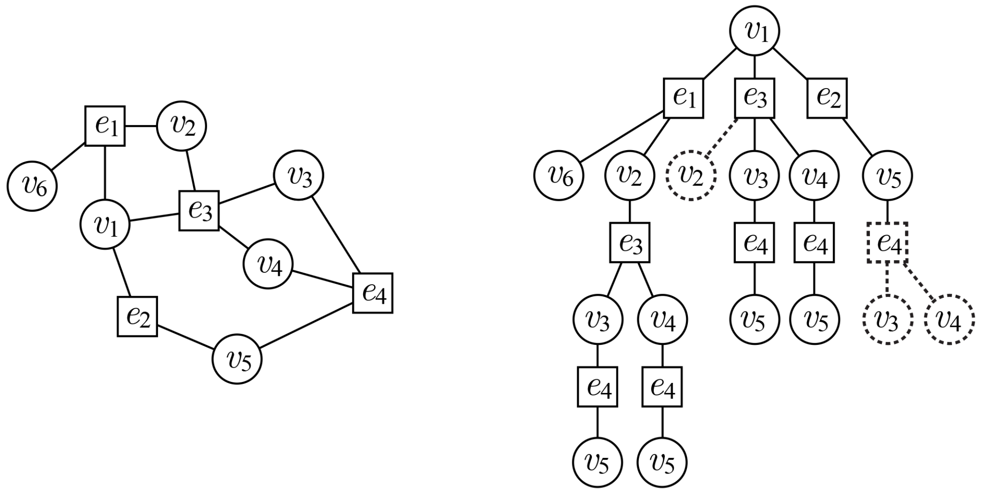


Figure 3 The construction of $\mathcal{T}_{\text{SAW}}$. On the left is a hypergraph $\mathcal{H}$ and on the right is $\mathcal{T}_{\text{SAW}}(\mathcal{H}, v_1)$, both drawn as incident graphs. The ordering of the hyperedges incident to each vertex in $\mathcal{H}$ is given by the subscripts. Each vertex or hyperedge in $\mathcal{T}_{\text{SAW}}$ is labeled by the name of the vertex or hyperedge to which it is identified in $\mathcal{H}$. Dashed vertices are the ones deleted according to the ordering of incident hyperedges at the cycle-closing vertices. Dashed hyperedge is deleted because its size becomes 1.

to u. With this local ordering of hyperedges, given any vertex $v \in V$, a rooted hypertree $\mathcal{T} = \mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$, called the *self-avoiding walk (SAW) tree*, is constructed as follows:

1. Every vertex of $\mathcal{T}$ corresponds to a distinct self-avoiding walk in $\mathcal{H}$ originating from v, where the root corresponds to the trivial walk (v).
2. For any vertex u in $\mathcal{T}$, which corresponds to a self-avoiding walk $w = (v, e_1, v_1 \dots, e_\ell, v_\ell)$, we partition all self-avoiding walks $w' = (v, e_1, v_1 \dots, e_\ell, v_\ell, e', v')$ in $\mathcal{H}$ which extends w, into sets according to which hyperedge they use to extend the original walk w, so that self-avoiding walks within the same sets extends w with the same hyperedge e'. For every set, we create a distinct hyperedge in $\mathcal{T}$ incident to u which contains the children of u corresponding to the self-avoiding walks within that set.
3. We further modify the hypertree $\mathcal{T}$ obtained from the above two steps according to how cycles are closed. For any vertex u in $\mathcal{T}$ corresponding to a self-avoiding walk $w = (v, e_1, v_1 \dots, e_\ell, v_\ell)$ which can be extended to a cycle-closing walk $w' = (v, e_1, v_1 \dots, e_\ell, v_\ell, e', v')$ such that $v' \in \{v, v_1, \dots, v_{\ell-1}\}$, denoted by e'' the hyperedge in w starting that cycle, if it holds that $e' <_{v'} e''$, i.e. the hyperedge ending the cycle is ranked higher than the hyperedge starting the cycle by the cycle-closing vertex, then vertex u along with all its descendants in $\mathcal{T}$ are deleted from $\mathcal{T}$. Any hyperedges whose size becomes 1 because of this step are also deleted from $\mathcal{T}$.

The construction is illustrated in Figure 3.

We consider the Gibbs measure of a rooted hypertree $\mathcal{T}$ with activity λ, and use $\mathbb{P}_{\mathcal{T}}^{\sigma_\Lambda}$ to denote the marginal probability of the root of $\mathcal{T}$ being occupied conditioning on σ_Λ.

Note that each vertex u in $\mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$ can be naturally identified (many-to-one) to the vertex in $\mathcal{H} = (V, E)$ at which the self-avoiding walk corresponding to u ends, thus a configuration σ_Λ partially specified on a subset $\Lambda \subset V$ of vertices in $\mathcal{H}$ can be directly translated to a partially specified configuration in $\mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$ through the one-to-many association. We abuse the notation and still denote the resulting configuration in $\mathcal{T} = \mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$ as σ_Λ, thus $\mathbb{P}_{\mathcal{T}}^{\sigma_\Lambda}$ is well-defined.

▶ **Theorem 15.** *Let $\mathcal{H} = (V, E)$ be a hypergraph and $\lambda > 0$. For any $v \in V$, $\Lambda \subseteq V$ and $\sigma_\Lambda \in \{0,1\}^\Lambda$, it holds that $p^{\sigma_\Lambda}_{\mathcal{H},v} = \mathbb{P}^{\sigma_\Lambda}_{\mathcal{T}}$ where $\mathcal{T} = \mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$.*

Proof. The proof follows the same routine as that of Weitz [34], with some extra cares to be taken to avoid the complications caused by hypergraphs.

Denote $R^{\sigma_\Lambda}_{\mathcal{H},v}(\lambda) = p^{\sigma_\Lambda}_{\mathcal{H},v}/(1 - p^{\sigma_\Lambda}_{\mathcal{H},v})$ for the ratio between the probability that v in $\mathcal{H}$ is occupied and unoccupied conditioning on configuration σ_Λ of $\Lambda \subset V$. We write $R^{\sigma_\Lambda}_{\mathcal{T}} = R^{\sigma_\Lambda}_{\mathcal{T},v}$ when v is unambiguously the root of $\mathcal{T}$.

Let d be the degree of the root of $\mathcal{T}$. Suppose that there are k_i children contained in i-th child-edge, where the order is determined during the construction of $\mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$. $\mathcal{T}_{ij}$ is the subtree rooted at the j-th child in the i-th child-edge. Let $\Lambda_{ij} = \Lambda \cap \mathcal{T}_{ij}$ and $\sigma_{\Lambda_{ij}}$ be the restriction of σ_Λ on Λ_{ij}. Applying the tree recursion (3) for the self-avoiding walk tree $\mathcal{T}$, we have

$$R^{\sigma_\Lambda}_{\mathcal{T}} = \lambda \prod_{i=1}^{d} \frac{1}{1 + \sum_{i=1}^{k_i} R^{\sigma_{\Lambda_{ij}}}_{\mathcal{T}_{ij}}}, \tag{4}$$

This defines a recursive procedure for calculating $R^{\sigma_\Lambda}_{\mathcal{T}}$. The base cases are naturally defined when v lies in Λ, in which case $R^{\sigma_\Lambda}_{\mathcal{T}} = 0$ if v is fixed unoccupied or $R^{\sigma_\Lambda}_{\mathcal{T}} = \infty$ if it is fixed occupied, or when v has no child, in which case $R^{\sigma_\Lambda}_{\mathcal{T}} = \lambda$.

In the following we describe our procedure for calculating $R^{\sigma_\Lambda}_{\mathcal{H},v}$ at v in the original hypergraph $\mathcal{H}$. The problem comes that the ratio at different neighbors of v may still depend on each other when we fix the value at v since there may exist cycles in $\mathcal{H}$. We resolve this problem by editing the original hypergraph around v and imposing appropriate conditions for each neighbor of v.

Let $\mathcal{H}^v$ be the same hypergraph as $\mathcal{H}$ except that vertex $v \in V$ is substituted by d vertices $v_1, v_2, ..., v_d$, where d is the degree of v. Each vertex v_i is contained into a single hyperedge e_i, where e_i is the i-th hyperedge connecting v, and the order here is the same as the one determined in the definition of $\mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$. At the same time, we associated each v_i with an activity of $\lambda^{1/d}$ rather than λ. It is now clear to see that an independent set in $\mathcal{H}$ with v occupied has the same weight as the corresponding independent set in $\mathcal{H}^v$ with all the v_i occupied, and so is the case when v is unoccupied. Therefore, $R^{\sigma_\Lambda}_{\mathcal{H},v}$ equals to the ratio between the probabilities in $\mathcal{H}^v$ with all v_i $(1 \le i \le d)$ being occupied and unoccupied, conditioning on σ_Λ. Let τ_i be the configuration for vertex v_i in which the values of v_j are fixed to occupied if $j < i$ and unoccupied if $j > i$. We can then write this in a form of telescopic product:

$$R^{\sigma_\Lambda}_{\mathcal{H},v} = \prod_{i=1}^{d} R^{\sigma_\Lambda \tau_i}_{\mathcal{H}^v, v_i},$$

where $\sigma_\Lambda \tau_i$ means the combination of the two configurations σ_Λ and τ_i.

We can obtain the value of $R^{\sigma_\Lambda \tau_i}_{\mathcal{H}^v, v_i}$ by further fix vertices in e_i, the hyperedge containing v_i. Since now v_i is contained only in e_i, we can see that

$$R^{\sigma_\Lambda \tau_i}_{\mathcal{H}^v, v_i} = \frac{\lambda^{1/d}}{1 + \sum_{j=1}^{k_i} R^{\sigma_\Lambda \tau_i \rho_{ij}}_{\mathcal{H}^v / v_i, u_{ij}}},$$

where k_i is the number of the vertices other than v_i which is incident to e_i and ρ_{ij} is the configuration at vertices of e_i in which all the vertices $u_{ij'}$ other than u_{ij} are fixed to unoccupied.

Combining above two equations, we get a recursive procedure for calculating $R^{\sigma_\Lambda}_{\mathcal{H},v}$ in the same manner that equation (4) has:

$$R^{\sigma_\Lambda}_{\mathcal{H},v} = \lambda \prod_{i=1}^{d} \frac{1}{1 + \sum_{j=1}^{k_i} R^{\sigma_\Lambda \tau_i \rho_{ij}}_{\mathcal{H}^v/v_i,u_{ij}}}. \tag{5}$$

Notice that the recursion does terminate, since the number of unfixed vertices reduces at least by one in each step because in calculating $R^{\sigma_\Lambda \tau_i \rho_{ij}}_{\mathcal{H}^v/v_i,u_{ij}}$ all copies $v_{i'}$ of v is either fixed (when $i' \neq i$) or erased (when $i' = i$) from the hypergraph $\mathcal{H}^v/v_i$.

We now show that the procedure described above for calculating $R^{\sigma_\Lambda}_{\mathcal{H},v}$ results in the same value as using the hypertree procedure for $\mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$ with corresponding condition of σ_Λ imposed on it. First notice that the calculation carried out by the two procedure is the same, since they share the same function (Equation (4) and (5)) when we view them as recursive calls. Furthermore, we have the same stopping values for the both recursive procedures. During constructing $\mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$, if node u corresponding to walk is not included in the hypertree, which is equivalent to fix u to unoccupied in the sense of causing the same effect on the ratio of occupation to its parent node. And when node u in the hypertree corresponding to a self-avoiding walk $w = (v, e_1, v_1 \ldots, e_\ell, v_\ell)$, with that w can be extended as $w' = (w, e_{\ell+1}, v_{\ell+1})$ to a cycle-closing vertex $v_{\ell+1} = v_i$ for some $0 \le i < \ell$ via a new hyperedge $e_{\ell+1} \notin \{e_0, e_1, \ldots, e_\ell\}$, and $e_{\ell+1} <_{v_i} e_i$, then the node u along with all its descendants are deleted. This gives the equivalent effect to parent node of u as if u is fixed to unoccupied, or one of the children of u (i.e. the node corresponding to w') to occupied, which is what we did to fix the vertices v_j for $j < i$ in τ_i. Eliminating a hyperedge with no child also does not affect the final value of $R^{\sigma_\Lambda}_{\mathcal{T}}$.

Thus, what is left to complete the proof is to show that the hypertree $\mathcal{T}_{\text{SAW}}(\mathcal{H}^v/v_i, u_{ij})$ with $(\sigma_\Lambda \tau_i \rho_{ij})$'s corresponding condition imposed on it is exactly the same as the subtree of $\mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$ rooted at the j-th child vertex of the i-th child-edge of the root with σ_Λ's corresponding condition imposed on it. This is enough because then the resulting values are the same for both procedures by induction. The observation is that both trees are the hypertree of all self-avoiding walks in $\mathcal{H}$ starting at u_{ij}, except that $\mathcal{T}_{\text{SAW}}(\mathcal{H}^v/v_i, u_{ij})$ has some extra vertices which are fixed to be occupied or unoccupied depending on whether the corresponding walk reaches v via a higher or lower ranked hyperedge, or reaches i-th hyperedge of v, which results in the same probability of occupation at the root. ◀

A hypergraph $\mathcal{H}$ is a sub-hypergraph of another hypergraph $\mathcal{G}$ if the incidence graph of $\mathcal{H}$ is a subgraph of that of $\mathcal{G}$, and for hypertrees this is samely defined. Note that for hypergraphs, a subgraph is not necessarily formed by a sub-collection of hyperedges, but maybe also by sub-hyperedges. The $\mathcal{T}_{\text{SAW}}$ of a hypergraph $\mathcal{H}$ with maximum edge-size at most $k + 1$ and maximum degree at most $d + 1$ is sub-hypertree of $\mathbb{T}_{k,d}$.

▶ **Proposition 16.** *Let $\mathcal{T}_0 = (V_0, E_0)$ be a rooted hypertree and $\mathcal{T} = (V, E)$ its sub-hypertree with the same root. For any $\Lambda \subseteq V$ and any $\sigma_\Lambda \in \{0, 1\}^\Lambda$, there exists a configuration $\sigma_{\Lambda_0} \in \{0, 1\}^{\Lambda_0}$ for $\Lambda \subseteq \Lambda_0 \subseteq V_0$, extending the configuration σ_Λ, such that $\mathbb{P}^{\sigma_\Lambda}_{\mathcal{T}} = \mathbb{P}^{\sigma_{\Lambda_0}}_{\mathcal{T}_0}$.*

The configuration σ_{Λ_0} just extends σ_Λ by fixing all the vertices missing in $\mathcal{T}$ (actually only those who are closest to the root along each path) to be unoccupied.

Theorem 14 follows immediately from Theorem 15 and Proposition 16.

Proof of Theorem 14. Given any hypergraph $\mathcal{H}$ of maximum edge-size at most $(k + 1)$ and maximum degree at most $(d + 1)$, by Theorem 15 we have $|p^{\sigma_\Lambda}_{\mathcal{H},v} - p^{\tau_\Lambda}_{\mathcal{H},v}| = |\mathbb{P}^{\sigma_\Lambda}_{\mathcal{T}} - \mathbb{P}^{\tau_\Lambda}_{\mathcal{T}}|$ where

$\mathcal{T} = \mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$. The distance from the root v to any vertex u in $\mathcal{T}$ is no shorter than the distance $\mathcal{H}$ between v and the vertex in $\mathcal{H}$ to which u is identified. So the SSM with rate $\delta(\cdot)$ on $\mathcal{T}$ implies that on the hypergraph $\mathcal{H}$.

Since $\mathcal{H}$ has maximum edge-size at most $k+1$ and maximum degree at most $d+1$, its SAW-tree $\mathcal{T} = \mathcal{T}_{\text{SAW}}(\mathcal{H}, v)$ is a sub-hypertree of $\mathbb{T}_{k,d}$. Thus by Proposition 16, we have $|\mathbb{P}_{\mathcal{T}}^{\sigma_\Lambda} - \mathbb{P}_{\mathcal{T}}^{\tau_\Lambda}| = |\mathbb{P}_{\mathbb{T}_{k,d}}^{\sigma_{\Lambda_0}} - \mathbb{P}_{\mathbb{T}_{k,d}}^{\tau_{\Lambda_0}}|$ for some $\sigma_{\Lambda_0}, \tau_{\Lambda_0}$ extending $\sigma_\Lambda, \tau_\Lambda$. The SSM on $\mathbb{T}_{k,d}$ with rate $\delta(\cdot)$ implies that on $\mathcal{T}$, which implies the same on the original hypergraph $\mathcal{H}$. ◀

5 Strong spatial mixing

In this section, we show that for independent sets of the infinite $(k+1)$-uniform $(d+1)$-regular hypertree $\mathbb{T}_{k,d}$, weak spatial mixing implies strong spatial mixing at almost the same rate.

▶ **Theorem 17.** *For every positive integers d, k and any λ, if the independent sets of the infinite $(k+1)$-uniform $(d+1)$-regular hypertree $\mathbb{T}_{k,d}$ with activity λ exhibits weak spatial mixing with rate $\delta(\cdot)$ then it also exhibits strong spatial mixing with rate $\frac{(1+\lambda)\left(\lambda+(1+k\lambda)^{d+1}\right)}{\lambda}\delta(\cdot)$.*

By Theorem 14, this implies the strong spatial mixing with the same rate on all hypergraphs of maximum degree at most $d+1$ and maximum size of hyperedges at most $k+1$.

Unlike most known strong spatial mixing results, where the spatial mixing is usually established by an analytic approach with help of potential functions, our proof of Theorem 17 adopts the combinatorial argument used in Weitz's original proof of SSM for the hardcore model [34]. Weitz's approach gives us a stronger result: It explicitly gives the extremal case for WSM as well as SSM among a family of hypergraphs with bounded maximum degree and bounded maximum edge-size. It can also easily give us the SSM behavior when at the critical threshold.

Assume the hypertree $T = \mathbb{T}_{k,d}$ is rooted at some vertex v. For $\ell > 0$, let R_ℓ^+ and R_ℓ^- denote the respective maximum and minimum values of R_T^σ achieved by a boundary condition σ that fixes the states of all vertices at level ℓ. By the monotonicity of the tree recursion, it is easy to see that R_ℓ^+ (or R_ℓ^-) is computed by the tree recursion with initial values at all vertices at level ℓ to be ∞ (or 0) if ℓ is even, and 0 (or ∞) if ℓ is odd, with the root v being at level 0.[2]

It is easy to see that fixing a vertex u in T to be occupied has the same effect as fixing u's parent to be unoccupied, therefore to prove SSM, it is sufficient to prove the decay of correlation conditioning on a subset of vertices in T fixed to be unoccupied. Another key observation from the tree recursion is that fixing a vertex u in T to be unoccupied has the same effect as having a local activity $\lambda_u = 0$ at vertex u. Now consider a vector $\vec{\lambda}$ that assigns every vertex u in $T = \mathbb{T}_{k,d}$ a local activity λ_u. Let $R_\ell^+(\vec{\lambda})$ and $R_\ell^-(\vec{\lambda})$ be accordingly defined as the respective extremal values of $R_T^\sigma(\vec{\lambda})$ achieved by boundary conditions σ fixing all vertices at level ℓ in the tree $T = \mathbb{T}_{k,d}$ equipped with the nonuniform activities $\vec{\lambda}$. Clearly, by the same monotonicity, $R_\ell^\pm(\vec{\lambda})$ can be computed from the tree recursion with a nonuniform activities $\vec{\lambda}$ with the same settings of initial values as the uniform case $R_\ell^\pm = R_\ell^\pm(\lambda)$.

The following theorem shows that basically the decay of correlation is dominated by the uniform activity case.

[2] Note that although the all-∞ initial values corresponds to a boundary condition σ that fixes all vertices at level ℓ to be occupied, which may no longer be a valid independent set in the hypertree, the $R_\ell^\pm$ achieved by this choice of initial values is actually the same as the R_T^σ with a boundary condition σ that fixes exactly one vertex per hyperedge to be occupied at level ℓ.

▶ **Theorem 18.** *Fix an arbitrary $\lambda \geq 0$. Let $\vec{\lambda}$ be an assignment of activities to vertices of $\mathbb{T}_{k,d}$ such that $0 \leq \lambda_v \leq \lambda$ for every $v \in \mathbb{T}_{k,d}$. For every $\ell \geq 1$ we have*

$$\frac{R_\ell^+(\vec{\lambda})}{R_\ell^-(\vec{\lambda})} \leq \frac{R_\ell^+}{R_\ell^-}.$$

Translated to the language of subtrees, the theorem means that the extremal case of WSM among a family hypertrees with bounded maximum degree and maximum edge-size, is given by the uniform regular tree with the highest degree and edge-size in the family. Technically, Theorem 18 measures the decay of correlation in terms of $\log R = \log \frac{p}{1-p}$. Note that for $\ell \geq 2$, it always holds that $p_\ell^+(\lambda) \leq \frac{\lambda}{1+\lambda}$ and $p_\ell^-(\lambda) \geq \frac{\lambda}{\lambda+(1+k\lambda)^{d+1}}$, where $R_\ell^\pm = \frac{p_\ell^\pm}{1-p_\ell^\pm}$. Theorem 18 implies Theorem 17.

We now consider a slightly different hypertree which is exactly the same as $\mathbb{T}_{k,d}$ except that the degree of root is d. Denote this hypertree as $\widehat{\mathbb{T}}_{k,d}$.

▶ **Lemma 19.** *For every integer $\ell \geq 1$ and any assignment of activities $\vec{\lambda}$ to vertices of $\widehat{\mathbb{T}}_{k,d}$ such that $0 \leq \lambda_v \leq \lambda$ for every vertex v, the following two inequalities hold:*

$$\frac{R_\ell^+(\vec{\lambda})}{R_\ell^-(\vec{\lambda})} \leq \frac{R_\ell^+}{R_\ell^-}, \tag{6}$$

$$\frac{1+kR_\ell^+(\vec{\lambda})}{1+kR_\ell^-(\vec{\lambda})} \leq \frac{1+kR_\ell^+}{1+kR_\ell^-}, \tag{7}$$

with the convention $0/0 = 1$ and $\infty = \infty$.

Proof of Lemma 19. The proof is by an induction on ℓ. The proof is similar to that of Weitz [34] except for the parts dealing with hyperedges.

First consider the exceptional cases when the denominators in (6) may be zero. Assume $R_\ell^+(\vec{\lambda}) = R_\ell^-(\vec{\lambda}) = 0$, which only happens when the activity of the root is zero. We adopt the convention that $\frac{R_\ell^+(\vec{\lambda})}{R_\ell^-(\vec{\lambda})} = 1$. Assume $R_\ell^- = 0$, which only occurs when $\ell = 1$. Then $R_\ell^-(\vec{\lambda}) = 0$ also holds, and by convention we have $\frac{R_\ell^+(\vec{\lambda})}{R_\ell^-(\vec{\lambda})} = \frac{R_\ell^+}{R_\ell^-} = \infty$. Note that these conventions are consistent with our induction, such that assuming the induction hypothesis $\frac{R_\ell^+(\vec{\lambda})}{R_\ell^-(\vec{\lambda})} \leq \frac{R_\ell^+(\lambda)}{R_\ell^-(\lambda)}$, for any k assignments of activities $0 \leq \vec{\lambda_1}, \vec{\lambda_2}, ..., \vec{\lambda_k} \leq \lambda$, there exists $\alpha \geq 0$ such that $\sum_{i=1}^k R_\ell^-(\vec{\lambda_i}) = \alpha k R_\ell^-$ and $\sum_{i=1}^k R_\ell^+(\vec{\lambda_i}) \leq \alpha k R_\ell^+$.

For the basis, $\ell = 1$. We have $R_\ell^-(\vec{\lambda}) \geq R_\ell^- = 0$, $R_\ell^+ = \lambda$, and $R_\ell^+(\vec{\lambda}) = \lambda_r$ where λ_r is the activity of the root. The hypotheses (6) and (7) are true since $\lambda_r \leq \lambda$.

Assume (6) and (7) are true for an $\ell \geq 1$. We will show that they are true for $\ell + 1$. The following recursion holds

$$\frac{R_{\ell+1}^+(\vec{\lambda})}{R_{\ell+1}^-(\vec{\lambda})} = \prod_{i=1}^d \frac{1+\sum_{j=1}^k R_\ell^+(\vec{\lambda}_{ij})}{1+\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{ij})} = \prod_{i=1}^d \frac{\sum_{j=1}^k (1+kR_\ell^+(\vec{\lambda}_{ij}))}{\sum_{j=1}^k (1+kR_\ell^-(\vec{\lambda}_{ij}))},$$

where $\vec{\lambda}_{ij}$ stands for the restriction of the assignment $\vec{\lambda}$ to the subtree of $\mathbb{T}_{k,d}$ rooted at the j-th child in the i-th edge incident to the root. By induction hypothesis (7), we have $\frac{1+kR_\ell^+(\vec{\lambda}_{ij})}{1+kR_\ell^-(\vec{\lambda}_{ij})} \leq \frac{1+kR_\ell^+}{1+kR_\ell^-}$, so immediately,

$$\frac{R_{\ell+1}^+(\vec{\lambda})}{R_{\ell+1}^-(\vec{\lambda})} = \prod_{i=1}^d \frac{\sum_{j=1}^k (1+kR_\ell^+(\vec{\lambda}_{ij}))}{\sum_{j=1}^k (1+kR_\ell^-(\vec{\lambda}_{ij}))} \leq \left(\frac{1+kR_\ell^+}{1+kR_\ell^-}\right)^d = \frac{R_{\ell+1}^+}{R_{\ell+1}^-},$$

where the inequality is due to the simple fact that if $a_i \ge b_i > 0$ and $\frac{a_i}{b_i} \le t$ for all i, then $\frac{\sum_{i=1}^n a_i}{\sum_{i=1}^n b_i} \le t$.

This proves (6) for $\ell+1$. Next we will prove (7). Recall the tree recursions:

$$R_{\ell+1}^{\pm}(\vec{\lambda}) = \lambda_r \prod_{i=1}^{d} \frac{1}{1+\sum\limits_{j=1}^{k} R_\ell^{\mp}(\vec{\lambda}_{ij})} \quad \text{and} \quad R_{\ell+1}^{\pm} = \lambda \prod_{i=1}^{d} \frac{1}{1+kR_\ell^{\mp}},$$

where λ_r is the local activity assigned by $\vec{\lambda}$ to the root r. Observe that if $\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{ij}) \ge kR_\ell^-$ for all $i \in [d]$, then $R_{\ell+1}^+(\vec{\lambda}) \le R_{\ell+1}^+$, which combined with (6) for $\ell+1$ that we just proved above, would give us that $\frac{1+kR_{\ell+1}^+(\vec{\lambda})}{1+kR_{\ell+1}^-(\vec{\lambda})} \le \frac{1+kR_{\ell+1}^+}{1+kR_{\ell+1}^-}$. In this good case, the hypothesis (7) easily holds for $\ell+1$. We then show that the opposite case where $\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{ij}) \le kR_\ell^-$ for all i represents the worst possible case, and it is enough to prove the hypothesis (7) under this condition. To see this, assume to the contrary that for some i_0, $\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{i_0 j}) > kR_\ell^-$. We then construct a $\vec{\lambda}'$ that satisfies $\sum_{j=1}^k R_\ell^-(\vec{\lambda}'_{i_0 j}) \le kR_\ell^-$ and has an even worse ratio between $1+kR_{\ell+1}^+$ and $1+kR_{\ell+1}^-$. Let $\vec{\lambda}'$ be the same as $\vec{\lambda}$ except that for every $j \in [k]$, $\vec{\lambda}'_{i_0 j}$ is uniform and is equal to λ everywhere. Clearly, it holds that $\sum_{j=1}^k R_\ell^-(\vec{\lambda}'_{i_0 j}) = kR_\ell^-$. On the other hand, by the induction hypothesis, for every j we have $\frac{1+kR_\ell^+(\vec{\lambda}_{i_0 j})}{1+kR_\ell^-(\vec{\lambda}_{i_0 j})} \le \frac{1+kR_\ell^+}{1+kR_\ell^-}$, and hence

$$\frac{1+\sum_{j=1}^k R_\ell^+(\vec{\lambda}_{i_0 j})}{1+\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{i_0 j})} = \frac{\sum_{j=1}^k (1+kR_\ell^+(\vec{\lambda}_{i_0 j}))}{\sum_{j=1}^k (1+kR_\ell^-(\vec{\lambda}_{i_0 j}))} \le \frac{1+kR_\ell^+}{1+kR_\ell^-},$$

where again the inequality uses the fact that if $a_i \ge b_i > 0$ and $\frac{a_i}{b_i} \le t$ for all i, then $\frac{\sum_{i=1}^n a_i}{\sum_{i=1}^n b_i} \le t$.

Note that $\vec{\lambda}'$ only changes the activities of all the subtrees rooted by the the children in the i_0-th edge of the root. So we have

$$\begin{aligned} \frac{R_{\ell+1}^+(\vec{\lambda})}{R_{\ell+1}^-(\vec{\lambda})} &= \prod_{i=1}^{d} \frac{1+\sum_{j=1}^k R_\ell^+(\vec{\lambda}_{ij})}{1+\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{ij})} \le \frac{R_{\ell+1}^+(\vec{\lambda}')}{R_{\ell+1}^-(\vec{\lambda}')}, \\ \text{and} \quad R_{\ell+1}^+(\vec{\lambda}) &= \lambda_r \prod_{i=1}^{d} \frac{1}{1+\sum\limits_{j=1}^{k} R_l^-(\vec{\lambda}_{ij})} \le R_{l+1}^+(\vec{\lambda}'). \end{aligned}$$

Combine the two inequalities, we have $\frac{1+kR_{\ell+1}^+(\vec{\lambda})}{1+kR_{\ell+1}^-(\vec{\lambda})} \le \frac{1+kR_{\ell+1}^+(\vec{\lambda}')}{1+kR_{\ell+1}^-(\vec{\lambda}')}$, an even worse case. So for the rest we only need to consider the case in which for every i, $\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{ij}) \le kR_\ell^-$.

For every $1 \le i \le d$, we can choose $0 \le \alpha_i \le 1$ so that $\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{ij}) = \alpha_i kR_\ell^-$. Fix i and by the induction hypothesis, for every $1 \le j \le k$ we have $\frac{R_\ell^+(\vec{\lambda}_{ij})}{R_\ell^-(\vec{\lambda}_{ij})} \le \frac{R_\ell^+}{R_\ell^-}$. If all $R_\ell^-(\vec{\lambda}_{ij})$ equal zero, then $\sum_{j=1}^k R_\ell^+(\vec{\lambda}_{ij}) \le \alpha_i kR_\ell^+$ trivially holds as we argued in the beginning. Otherwise, note that since not all $R_\ell^-(\vec{\lambda}_{ij})$ are zero, we must have $\ell > 1$, so if $R_\ell^-(\vec{\lambda}_{ij}) = 0$ then $R_\ell^+(\vec{\lambda}_{ij}) = 0$. Thus we also have $\frac{\sum_{j=1}^k R_\ell^+(\vec{\lambda}_{ij})}{\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{ij})} \le \frac{R_\ell^+}{R_\ell^-}$. In conclusion, in both cases we have $\sum_{j=1}^k R_\ell^+(\vec{\lambda}_{ij}) \le \alpha_i kR_\ell^+$.

Observe that $\lambda_r \le \lambda$ and $\prod_{i=1}^d \frac{1+\sum_{j=1}^k R_\ell^+(\vec{\lambda}_{ij})}{1+\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{ij})} \ge 1$, it holds that

$$\frac{1+kR_{\ell+1}^+(\vec{\lambda})}{1+kR_{\ell+1}^-(\vec{\lambda})} = \frac{1+k\lambda_r\prod_{i=1}^d \frac{1}{1+\sum_{j=1}^k R_\ell^-(\vec{\lambda}_{ij})}}{1+k\lambda_r\prod_{i=1}^d \frac{1}{1+\sum_{j=1}^k R_\ell^+(\vec{\lambda}_{ij})}} \le \frac{1+k\lambda\prod_{i=1}^d \frac{1}{1+\alpha_i kR_l^-}}{1+k\lambda\prod_{i=1}^d \frac{1}{1+\alpha_i kR_\ell^+}}.$$

Now it is enough to show that for every $\vec{\alpha}$ such that $0 \le \alpha_i \le 1$ for all $1 \le i \le d$, it holds that

$$\frac{1+k\lambda\prod_{i=1}^d \frac{1}{1+\alpha_i kR_l^-}}{1+k\lambda\prod_{i=1}^d \frac{1}{1+\alpha_i kR_\ell^+}} \le \frac{1+kR_{\ell+1}^+}{1+kR_{\ell+1}^-},$$

which is equivalent to the following:

$$1+k\lambda\prod_{i=1}^d \frac{1}{1+\alpha_i kR_\ell^-} - \frac{1+kR_{\ell+1}^+}{1+kR_{\ell+1}^-} - k\lambda\frac{1+kR_{\ell+1}^+}{1+kR_{\ell+1}^-}\prod_{i=1}^d \frac{1}{1+\alpha_i kR_\ell^+} \le 0. \tag{8}$$

If $\alpha_i = 1$ for every i then the inequality (8) trivially holds. By symmetry, we only need to show the LHS of (8) is increasing in α_1. In fact, the partial derivative with respect to α_1 of LHS in (8) is:

$$-\frac{k^2\lambda R_\ell^-}{1+\alpha_1 kR_\ell^-}\prod_{i=1}^d \frac{1}{1+\alpha_i kR_\ell^-} + \frac{k^2\lambda\left(1+kR_{\ell+1}^+\right)R_\ell^+}{\left(1+kR_{\ell+1}^-\right)\left(1+\alpha_1 kR_\ell^+\right)}\prod_{i=1}^d \frac{1}{1+\alpha_i kR_\ell^+}.$$

To prove it is nonnegative, it is equivalent to show that

$$\frac{\left(1+kR_{\ell+1}^+\right)R_\ell^+}{\left(1+kR_{\ell+1}^-\right)R_\ell^-} \ge \frac{1+\alpha_1 kR_\ell^+}{1+\alpha_1 kR_\ell^-}\prod_{i=1}^d \frac{1+\alpha_i kR_\ell^+}{1+\alpha_i kR_\ell^-}. \tag{9}$$

To prove (9), we first observe that R_ℓ^- is increasing in ℓ and R_ℓ^+ is decreasing in ℓ, which is exactly the same to prove as the same property of the hardcore model. This gives us the so-called sandwich condition:

$$R_\ell^- \le R_{\ell+1}^- \le R_{\ell+1}^+ \le R_\ell^+,$$

therefore $\frac{R_\ell^+}{R_\ell^-} \ge \frac{R_{\ell+1}^+}{R_{\ell+1}^-}$. We are now ready to prove (9):

$$\begin{aligned}\frac{\left(1+kR_{\ell+1}^+\right)R_\ell^+}{\left(1+kR_{\ell+1}^-\right)R_\ell^-} \ge& \frac{\left(1+kR_\ell^+\right)R_{\ell+1}^+}{\left(1+kR_\ell^-\right)R_{\ell+1}^-}\\ =& \frac{1+kR_\ell^+}{1+kR_\ell^-}\prod_{i=1}^d \frac{1+kR_\ell^+}{1+kR_\ell^-}\\ \ge& \frac{1+\alpha_1 kR_\ell^+}{1+\alpha_1 kR_\ell^-}\prod_{i=1}^d \frac{1+\alpha_i kR_\ell^+}{1+\alpha_i kR_\ell^-}.\end{aligned}$$

The last inequality uses the fact that $R_\ell^+ \ge R_\ell^-$ and $0 \le \alpha_i \le 1$. So (9) is proved, which finishes our proof of Lemma 19. ◀

Observe that the subtree rooted at the child of the root of $\mathbb{T}_{k,d}$ is isomorphic to $\widehat{\mathbb{T}}_{k,d}$. While at the root of $\mathbb{T}_{k,d}$, we have

$$\frac{R_\ell^+(\vec{\lambda})}{R_\ell^-(\vec{\lambda})} = \prod_{i=1}^{d+1} \frac{1+\sum_{j=1}^k R_{\ell-1}^+(\vec{\lambda}_{ij})}{1+\sum_{j=1}^k R_{\ell-1}^-(\vec{\lambda}_{ij})} \le \left(\frac{1+kR_{\ell-1}^+}{1+kR_{\ell-1}^-}\right)^{d+1} = \frac{R_\ell^+(\mathbb{T}_{k,d})}{R_\ell^-(\mathbb{T}_{k,d})}.$$

Together with Lemma 19, this completes our proof of Theorem 18.

Calculation of the decay rate

The WSM rate of our model on the infinite $(k+1)$-uniform $(d+1)$-regular hypertree $\mathbb{T}_{k,d}$ is the same as the hardcore model on the infinite $(d+1)$-regular tree with activity $k\lambda$. The WSM rate on regular tree has been addressed implicitly in the literature [18, 32]. Here we provide an analysis for the decay rate for the completeness of the paper.

Let $f_{d,k}(x) \triangleq \frac{k\lambda}{(1+x)^d}$ denote the symmetric version of the tree recursion on $\widehat{\mathbb{T}}_{k,d}$ and substituting $x = kR$. Since $f_{d,k}(x)$ is decreasing in x, it follows that there is a unique positive fixed point $\hat{x}$ such that $\hat{x} = f_{d,k}(\hat{x})$. Let $f'_{d,k}(\hat{x}) = -\frac{d\hat{x}}{1+\hat{x}}$ be the derivative of $f_{d,k}(x)$ evaluated at the fixed point $x = \hat{x}$. The following proposition is well known for hardcore model (see e.g. [18, 32]).

▶ **Proposition 20.** *$|f'_{d,k}(\hat{x})| = \frac{d\hat{x}}{1+\hat{x}} \le 1$ if and only if $\lambda \le \lambda_c$. And $|f'_{d,k}(\hat{x})| < 1$ if $\lambda < \lambda_c$.*

We write $f(x) = f_{d,k}(x)$ if k and d are clear in the context. The main result of this part is the following theorem.

▶ **Theorem 21.** *For any positive integers d, k, assuming $\lambda \le \lambda_c$, the model on $\mathbb{T}_{k,d}$ exhibits weak spatial mixing with rate $\delta(\ell)$ such that for all sufficiently large ℓ:*
- *if $\lambda < \lambda_c$, then $\delta(\ell) \le C_1|f'(\hat{x})|^{\ell-4}$;*
- *if $\lambda = \lambda_c$, then $\delta(\ell) \le \frac{C_2}{\sqrt{\ell-\ell_0}}$;*

where $C_1, C_2, \ell_0 > 0$ are finite constants depending only on k, d and λ.

Theorem 14, Theorem 17 and 21 together prove the SSM part of Theorem 2.

Denote $g(x) = f(f(x)) = k\lambda\left(1+\frac{k\lambda}{(1+x)^d}\right)^{-d}$. It is easy to see that $\hat{x} = g(\hat{x})$.

▶ **Lemma 22.** *If $\lambda \le \lambda_c$ then for any $x > \hat{x}$ we have $g(x) - g(\hat{x}) \le f'(\hat{x})^2(x-\hat{x})$.*

Proof. By the mean value theorem, for any $x > \hat{x}$, there exists a $z \in [\hat{x}, x]$ such that

$$g(x) - g(\hat{x}) = \alpha(z)(x-\hat{x}), \tag{10}$$

where $\alpha(z) = g'(z) = \frac{d^2k\lambda g(z)}{(1+z)^{d+1}+(1+z)k\lambda}$. We will bound the maximum value of $\alpha(z)$ when $\lambda \le \lambda_c$. Consider the derivative of $\alpha(z)$,

$$\alpha'(z) = A(z)\left[(d-1)k\lambda - (1+z)^d\right],$$

where $A(z) = \frac{d^2(d+1)k\lambda g(z)}{[(1+z)^{d+1}+(1+z)k\lambda]^2} > 0$. Let $z^* = ((d-1)k\lambda)^{1/d} - 1$ be the solution of $(d-1)k\lambda = (1+z)^d$. Note that $\left[(d-1)k\lambda - (1+z)^d\right]$ is decreasing in z, therefore $\alpha(z) \le \alpha(z^*)$ for all $z > 0$. Due to proposition 20, if $\lambda \le \lambda_c$ then $|f'(\hat{x})| = \frac{d\hat{x}}{1+\hat{x}} \le 1$ and hence $\hat{x} \le \frac{1}{d-1}$, thus $\alpha'(\hat{x}) = A(\hat{x})[(d-1)k\lambda - \frac{k\lambda}{\hat{x}}] \le 0$, which means $\hat{x} \ge z^*$ and $\alpha(z)$ is decreasing in z for any $z \ge \hat{x}$. On the other hand, we have $\alpha(\hat{x}) = f'(\hat{x})^2$. Thus for any $z \ge \hat{x}$, we have $\alpha(z) \le \alpha(\hat{x}) = f'(\hat{x})^2$. Plug it into (10). The lemma is proved. ◀

Proof of Theorem 21. It holds that $R_2^+ = R_1^+ = \lambda > \hat{x}/k$. Note that $kR_\ell^+ = g(kR_{\ell-2}^+)$. Due to the monotonicity of $g(x)$, we have $\hat{x} < kR_\ell^+ \le k\lambda$ for every $\ell \ge 1$.

Consider the case $\lambda < \lambda_c$. First consider the $(k+1)$-uniform d-ary hypertree $\widehat{\mathbb{T}}_{k,d}$. By the mean value theorem and Lemma 22 we have

$$kR_\ell^+ - \hat{x} = g(kR_{\ell-2}^+) - g(\hat{x}) \le f'(\hat{x})^2 \left(kR_{\ell-2}^+ - \hat{x}\right).$$

We apply this inequality recursively. Since $kR_\ell^+ - \hat{x} < k\lambda$, for any $\ell \ge 2$ we have

$$kR_\ell^+ - \hat{x} \le k\lambda |f'(\hat{x})|^{\ell-2}. \tag{11}$$

To bound R_ℓ^- we apply the mean value theorem again. There exists a $z \in [\hat{x}, kR_\ell^+]$ such that

$$\hat{x} - kR_\ell^- = f(\hat{x}) - f(kR_{\ell-1}^+) = |f'(z)|(kR_{\ell-1}^+ - \hat{x}).$$

Since $|f'(z)| \le kd\lambda$ for all $z > 0$, combined with (11) we have

$$\hat{x} - kR_\ell^- \le kd\lambda(kR_{\ell-1}^+ - \hat{x}) \le k^2 d\lambda^2 |f'(\hat{x})|^{\ell-3}.$$

At last, $R_\ell^+ - R_\ell^- = \frac{1}{k}(kR_\ell^+ - \hat{x} + \hat{x} - kR_\ell^-) \le C_1' |f'(\hat{x})|^{\ell-3}$ for some $C_1' > 0$ depending only on d, k and λ. This only gives us the desired decay rate at the $(k+1)$-uniform d-ary hypertree $\widehat{\mathbb{T}}_{k,d}$. Move to the $(k+1)$-uniform $(d+1)$-regular hypertree $\mathbb{T}_{k,d}$. The only difference is that the root has $d+1$ children instead of d. By the mean value theorem, this will multiply at most a finite constant factor C_1'' to the gap $R_\ell^+ - R_\ell^-$ at the root of $\mathbb{T}_{k,d}$, where $C_1'' > 0$ depends only on d, k and λ. Overall, this gives us that

$$p_\ell^+ - p_\ell^- \le R_\ell^+ - R_\ell^- \le C_1 |f'(\hat{x})|^{\ell-4}$$

for some $C_1 > 0$ depending only on d, k and λ. This finishes the case that $\lambda < \lambda_c$.

Now we consider the critical case that $\lambda = \lambda_c = \frac{d^d}{k(d-1)^{d+1}}$. We still start by considering the $(k+1)$-uniform d-ary hypertree $\widehat{\mathbb{T}}_{k,d}$. It is easy to verify that in this case $\hat{x} = \frac{1}{d-1}$, $\alpha(\hat{x}) = f'(\hat{x})^2 = 1$, $z^* = \hat{x}$ and $\alpha'(\hat{x}) = 0$, where $\alpha(z)$ and z^* are defined in the proof of Lemma 22. And we have $\alpha''(\hat{x}) = -\frac{(d+1)(d-1)^3}{d^2}$. By Taylor's expansion for $g(x)$ at the fixed point $x = \hat{x}$, we have that for any constant $c > 0$ there exists a constant $x_0 > \hat{x}$ such that for any $\hat{x} < x < x_0$, it holds that

$$\begin{aligned} g(x) &= g(\hat{x}) + \alpha(\hat{x})(x-\hat{x}) + \frac{\alpha'(\hat{x})}{2}(x-\hat{x})^2 + \frac{\alpha''(\hat{x})}{6}(x-\hat{x})^3 + o\left((x-\hat{x})^3\right) \\ &\le \frac{1}{d-1} + x - \hat{x} - \frac{(d+1)(d-1)^3}{6d^2}(x-\hat{x})^3 + c(x-\hat{x})^3. \end{aligned}$$

We define a sequence $x_1 = kR_1^+, x_3 = kR_3^+ = g(x_1), \ldots$ and generally $x_{2t+1} = g(x_{2t-1})$. The sequence is strictly decreasing because R_ℓ^+ is decreasing in ℓ. Furthermore, $\lim_{t\to\infty} x_{2t+1} = \hat{x}$. This is due to $\alpha(x) < 1$ for any $x > \hat{x}$.

Denote $\epsilon_{2t+1} \triangleq x_{2t+1} - \hat{x}$. Let c be some positive constant such that $\frac{(d+1)(d-1)^3}{6d^2} - c > 0$. Denote $\beta = \frac{(d+1)(d-1)^3}{6d^2} - c$ and $\gamma = \sqrt{\frac{1}{2\beta}}$. There must be some sufficiently large t_0 such that $\epsilon_{2t_0+1} \le \frac{\gamma}{\sqrt{2}}$ and for any $t > t_0$, it holds that

$$\epsilon_{2t+3} = g(x_{2t+1}) - \frac{1}{d-1} \le \epsilon_{2t+1} - \beta\epsilon_{2t+1}^3.$$

We apply an induction to complete the proof. For the basis, when $t = t_0$ we have $\epsilon_{2t_0+1} \le \frac{\gamma}{\sqrt{2}}$. Assume the hypothesis

$$\epsilon_{2t+1} \le \frac{\gamma}{\sqrt{t - t_0 + 2}} \tag{12}$$

for some $t \ge t_0$ and we will prove it holds for $t + 1$. First, notice that $h(x) \triangleq x - \beta x^3$ is strictly increasing when $0 \le x \le \frac{\gamma}{\sqrt{2}}$. Thus, we have

$$\epsilon_{2t+3} \le \epsilon_{2t+1} - \beta\epsilon_{2t+1}^3 \le \frac{\gamma}{\sqrt{t - t_0 + 2}} - \beta\frac{\gamma^3}{(t - t_0 + 2)^{\frac{3}{2}}}.$$

We only need to prove that $\frac{\gamma}{\sqrt{t-t_0+2}} - \beta\frac{\gamma^3}{(t-t_0+2)^{\frac{3}{2}}} \le \frac{\gamma}{\sqrt{t-t_0+3}}$. Let $t' \triangleq t - t_0 + 2$. It is equivalently to show that

$$t'^{\frac{3}{2}}\left(\frac{1}{\sqrt{t'}} - \frac{1}{\sqrt{t'+1}}\right) \le \beta\gamma^2. \tag{13}$$

Note that

$$\begin{aligned} t'^{\frac{3}{2}}\left(\frac{1}{\sqrt{t'}} - \frac{1}{\sqrt{t'+1}}\right) &= t'^{\frac{3}{2}}\left(\frac{\sqrt{t'+1} - \sqrt{t'}}{\sqrt{t'(t'+1)}}\right) \\ &\le \sqrt{t'}(\sqrt{t'+1} - \sqrt{t'}) \\ &\le \sqrt{t'}\frac{1}{\sqrt{t'+1} + \sqrt{t'}} \\ &\le \frac{1}{2}. \end{aligned}$$

Since $\beta\gamma^2 = \frac{1}{2}$, we just prove the inequality (13), and finishes the induction (12) for all $t \ge t_0$. In conclusion, for any $t \ge t_0$, it holds that

$$kR_{2t+1}^{+} - \hat{x} \le \frac{\gamma}{\sqrt{t - t_0 + 2}}.$$

The rest of the proof is exactly the same as our proof of the case $\lambda < \lambda_c$. ◀

6 Approximation algorithms and inapproximability

For $0 < \varepsilon < 1$, a value $\hat{Z}$ is an *ε-approximation* of Z if $(1 - \epsilon)Z \le \hat{Z} \le (1 + \epsilon)Z$. Recall that $\hat{x}$ is the unique fixed point solution to $\hat{x} = f_{d,k}(\hat{x}) = k\lambda(1 + \hat{x})^{-d}$.

▶ **Theorem 23.** *If $\lambda < \lambda_c = \frac{d^d}{k(d-1)^{d+1}}$, then there exists an algorithm such that given any $\varepsilon > 0$, and any hypergraph $\mathcal{H}$ of n vertices, of maximum edge-size at most $(k + 1)$ and maximum degree at most $(d + 1)$, the algorithm returns an ε-approximation of the partition function for the independent sets of $\mathcal{H}$ with activity λ, within running time $\left(\frac{n}{\varepsilon}\right)^{O\left(\frac{1}{\kappa}\ln kd\right)}$, where $\kappa = \ln\left(\frac{1+\hat{x}}{d\hat{x}}\right)$.*

For the critical case where $\lambda = \lambda_c$, there exists an algorithm that for the above $\mathcal{H}$ returns an ε-approximation of the log-partition function within running time $n(kd)^{O\left(\left(\frac{1}{\varepsilon}\ln\frac{1}{\varepsilon}\right)^2\right)}$.

By duality, the same algorithm with the same approximation ratio and running time works for the matchings of hypergraphs of maximum edge size at most $(d+1)$ and maximum degree at most $(k + 1)$. By Proposition 20, $|f'_{d,k}(\hat{x})| = \frac{d\hat{x}}{1+\hat{x}} < 1$ if $\lambda < \lambda_c$, therefore, when

$\lambda < \lambda_c$, the running time of the algorithm is $\mathrm{Poly}(n, \frac{1}{\epsilon})$ for any bounded k and d, so the algorithm is an FPTAS for the partition function. And when $\lambda = \lambda_c$, the algorithm is a PTAS for the log-partition function. The algorithmic part of the main theorem Theorem 2 is proved.

In particular, when $d = 1$, the model becomes matchings of graphs of maximum degree $(k+1)$, and the uniqueness condition $\lambda < \lambda_c(\mathbb{T}_{k,d})$ is always satisfied even for unbounded k since $\lambda_c(\mathbb{T}_{k,1}) = \infty$. In this case, the fixed point $\hat{x}$ for $f_{1,k}(x) = \frac{k\lambda}{1+x}$ can be explicitly solved as $\hat{x} = \frac{-1+\sqrt{1+4k\lambda}}{2}$. We have the following corollary for matchings of graphs with unbounded maximum degree, which achieves the same bound as the algorithm in [1].

▶ **Corollary 24.** *There exists an algorithm which given any graph G of maximum degree at most Δ, and any $\epsilon > 0$, returns an ε-approximation of the partition function for the matchings of G with activity λ, within running time $\left(\frac{n}{\epsilon}\right)^{O(\sqrt{\lambda\Delta}\log\Delta)}$.*

With the construction of hypergraph self-avoiding walk tree and the SSM, the algorithm follows the framework by Weitz [34]. We will describe an algorithm of approximating the partition function for independent sets in hypergraphs with activity λ. Under duality this is the same as approximately counting matchings with activity λ.

By the standard self-reduction, approximately computing the partition function is reduced to approximately computing the marginal probabilities. Let $\mathcal{H} = (V, E)$ be a hypergraph and $V = \{v_1, \ldots, v_n\}$. To calculate $Z = Z_{\mathcal{H}}(\lambda)$, it suffices to calculate the probability of the emptyset $\mu(\varnothing)$ as it is exactly $1/Z$. Let $\varnothing_i$ be the configuration on vertices v_1 up to v_i where all of them are unoccupied, and $p_{v_i}^{\varnothing_{i-1}}$ the probability of v_i being occupied conditioning on all vertices v_1 up to v_{i-1} being unoccupied. Then we have $1/Z = \prod_{i=1}^{n}(1 - p_{v_i}^{\varnothing_{i-1}})$ and $\log Z = -\sum_{i=1}^{n} \log(1 - p_{v_i}^{\varnothing_{i-1}})$. Note that $(1 - p_{v_i}^{\varnothing_{i-1}}) \geq \frac{1}{1+\lambda}$ for the probability of vertex unoccupied by an independent set and $\lambda_c \leq 4$ for any $d \geq 2$ and $k \geq 1$. To get an ε-approximation of Z, it suffices to approximate each of $p_{v_i}^{\varnothing_{i-1}}$ within an additive error $\frac{\varepsilon}{2(1+\lambda)n}$. And to get an ε-approximation of $\log Z$, which can be obtained by getting an ε-approximation of every $-\log(1 - p_{v_i}^{\varnothing_{i-1}})$, it is sufficient to approximate each of $p_{v_i}^{\varnothing_{i-1}}$ within an additive error $\Theta\left(\frac{\varepsilon}{\ln\frac{1}{\varepsilon}}\right)$.

By Theorem 14, we have $p_v^{\sigma} = \mathbb{P}_{\mathcal{T}}^{\sigma}$ where $\mathcal{T} = \mathcal{T}_{\mathrm{SAW}}(\mathcal{H}, v)$, i.e. the marginal probability of v being occupied is preserved in the SAW tree of $\mathcal{H}$ expanded at v. And the value of $\mathbb{P}_{\mathcal{T}}^{\sigma}$ can be computed by the tree recursion (3). To make the algorithm efficient we can run this recursion up to depth t and assume initial value 0 for the variables at depth t as the vertices they represent being unoccupied. The overall running time of the algorithm is clearly $O(n(kd)^t)$ where t is the depth of the recursion. By the strong spatial mixing guaranteed by Theorem 17 and Theorem 21, if $\lambda < \lambda_c$, then the additive error of such estimation of p_v^{σ} is bounded by $C_1 \cdot \left(\frac{d\hat{x}}{1+\hat{x}}\right)^{t-4}$ for some constant $C_1 > 0$ depending only on k, d and λ. We shall choose an integer t so that $C_1 \cdot \left(\frac{d\hat{x}}{1+\hat{x}}\right)^{t-4} \leq \frac{\varepsilon}{2(1+\lambda)n}$, which gives us the suitable time complexity required by the FPTAS for the partition function. And when $\lambda = \lambda_c$, the additive error of p_v^{σ} is bounded by $C_2/\sqrt{t - t_0}$ for some constants $C_2, t_0 > 0$ depending only on k, d. We shall choose an integer $t = O((\frac{1}{\varepsilon}\ln\frac{1}{\varepsilon})^2)$ to get the desirable additive error for every marginal probability, which gives us the PTAS for the log-partition function. This completes the proof of Theorem 23.

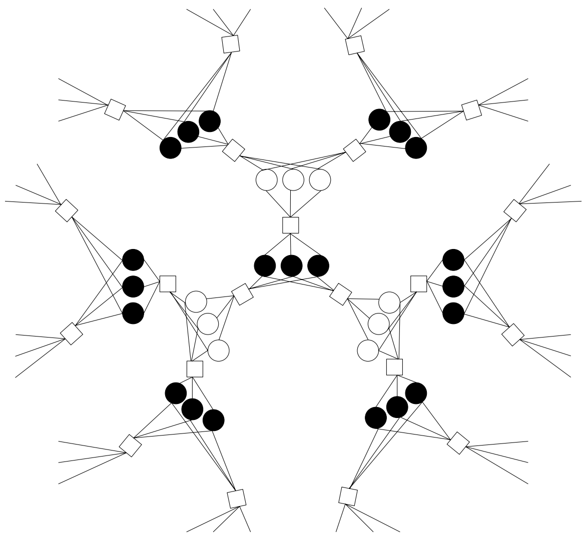


Figure 4 The infinite hypergraph that achieves the uniqueness threshold $\frac{2k+1+(-1)^k}{k+1}\lambda_c$.

Inapproximability

For the inapproximability, by applying an AP-reduction [3] from the inapproximability of the hardcore model [31, 9], we have the following theorem.

▶ **Theorem 25.** *If $\lambda > \frac{2k+1+(-1)^k}{k+1}\lambda_c$, there is no PRAS for the partition function or log-partition function of independent sets of hypergraphs with maximum degree at most $d+1$, maximum edge-size at most $k+1$ and activity λ, unless NP=RP.*

Proof. The reduction is as described in [3], which is reduced from the hardcore model. Given a graph $G(V,E)$ with maximum degree at most $(d+1)$, we construct a hypergraph $\mathcal{H}(V_{\mathcal{H}}, E_{\mathcal{H}})$ as follows. For each $v \in V$, we create $t = \left\lfloor \frac{k+1}{2} \right\rfloor$ distinct vertices $w_{v,1}, w_{v,2}, \ldots, w_{v,t}$ and let $V_{\mathcal{H}} = \{w_{v,i} \mid v \in V, 1 \le i \le t\}$. And for every edge $e = (u,v) \in E$, we create a hyperedge $S_e = \{w_{u,1}, \ldots, w_{u,t}, w_{v,1}, \ldots, w_{v,t}\}$ and let $E_{\mathcal{H}} = \{S_e \mid e \in E\}$. Clearly, the maximum degree of $\mathcal{H}$ is at most $d+1$ and the maximum edge-size of $\mathcal{H}$ is at most $2t \le k+1$. We define

$$Z_{\mathcal{H}}(\lambda) = \sum_{I:\ \text{IS of } \mathcal{H}} \lambda^{|I|} \quad \text{and} \quad Z_G(\lambda) = \sum_{I:\ \text{IS of } G} \lambda^{|I|}.$$

Note that by the above reduction every independent set I of $\mathbb{G}$ is naturally identified to $t^{|I|}$ distinct independent sets of hypergraph $\mathcal{H}$ such that a $v \in V$ is occupied by I if and only if one of $w_{v,i}$ is occupied by the corresponding independent set of $\mathcal{H}$. Thus $Z_{\mathcal{H}}(\lambda) = Z_G(\lambda')$ where $\lambda' = t\lambda$.

Recall that G is an arbitrary graph of maximum degree at most $d+1$. According to Sly and Sun [31], when $\lambda' > \frac{d^d}{(d-1)^{d+1}}$, there exists a constant c such that unless NP=RP, the partition function $Z_G(\lambda')$ can not be approximated within a factor of c^n in polynomial time, which means there is no PRAS for the log-partition function $\log Z_G(\lambda')$ when $\lambda' > \frac{d^d}{(d-1)^{d+1}}$, i.e. when $\lambda > \frac{d^d}{\lfloor (k+1)/2 \rfloor (d-1)^{d+1}} = \frac{2k+1+(-1)^k}{k+1}\lambda_c$. ◀

The reduction in Theorem 25 transforms a hardcore model on a graph with maximum degree $d+1$ and activity $\frac{2k+1+(-1)^k}{k+1}\lambda$ to an instance of hypergraph independent sets with maximum degree at most $d+1$, maximum edge-size at most $k+1$, and activity λ. In particular, it transforms the infinite $(d+1)$-regular tree $\mathbb{T}_{d,1}$ to the infinite $2\lfloor(k+1)/2\rfloor$-uniform hypergraph as shown in Figure 4. This infinite hypergraph has the uniqueness threshold $\frac{d^d}{\lfloor(k+1)/2\rfloor(d-1)^{d+1}} = \frac{2k+1+(-1)^k}{k+1}\lambda_c$.

7 Local convergence of hypergraphs

For the infinite $(k+1)$-uniform $(d+1)$-regular hypertree $\mathbb{T}_{k,d}$, a group $\mathbb{G}$ of automorphisms on $\mathbb{T}_{k,d}$ classifies the vertices and hyperedges in $\mathbb{T}_{k,d}$ into orbits (equivalent classes). We consider only $\mathbb{G}$ with finitely many orbits. By Proposition 11, such group $\mathbb{G}$ can be uniquely identified by a pair of branching matrices $(\boldsymbol{D},\boldsymbol{K})$ defined in Section 3 that classifies vertices and hyperedges in $\mathbb{T}_{k,d}$ into finitely many types (labels), where the incidence relation between vertices and hyperedges with each type is specified by $(\boldsymbol{D},\boldsymbol{K})$. We use $\mathbb{T}^{\mathbb{G}}_{k,d}$ to denote this resulting labeled hypertree.

For a finite hypergraph $\mathcal{H}=(V,E)$, we also consider the classification of vertices $V = \biguplus_{i\in[\tau_v]} V_i$ and hyperedges $E=\biguplus_{j\in[\tau_e]} E_j$ into disjoint types.

Given a hypergraph $\mathcal{H}$ and a vertex v in $\mathcal{H}$, write $B_t(v)=B_{\mathcal{H},t}(v)$ for the *t-neighborhood* around v in $\mathcal{H}$, that is, the sub-hypergraph induced by the vertices in $\mathcal{H}$ at distance at most t from v. For the labeled hypertree $\mathbb{T}^{\mathbb{G}}_{k,d}$, since once the type of the root is fixed the neighborhoods are identical (in terms of types), for each $i\in[\tau_v]$, we can denote $\mathbb{T}^{\mathbb{G}}_{k,d}(t,i)=B_{T,t}(v)$ where $T=\mathbb{T}^{\mathbb{G}}_{k,d}$ and v is any vertex in T of type-i.

The following definition is inspired by those of [31] and [4] for spin systems. Intuitively, a sequence of finite structures locally resemble the infinite tree structure along with the suitable symmetry which exhibits the uniqueness/nonuniqueness phase transition at the critical threshold, so the measures on the sequence of finite structures may have *local weak convergence* to that on the infinite tree. The existence of such local convergence profoundly leads to several most important phase-transition-based inapproximability results [6, 26, 30, 31, 7, 9, 10] and is a key to the success of random regular bipartite graph as a gadget for anti-ferromagnetic spin systems.

▶ **Definition 26** (local convergence). Let $\mathcal{H}_n=(V_n,E_n)$ be a sequence of random finite hypergraphs, whose vertices $V_n=\biguplus_{i\in[\tau_v]} V_{n,i}$ and hyperedges $E_n=\biguplus_{j\in[\tau_e]} E_{n,j}$ are classified into disjoint types, and for each $i\in[\tau_v]$, let $I_{n,i}\in V_{n,s}$ denote a uniformly random vertex in V_n of type-i.

We say the $\mathcal{H}_n$ *converge locally* to $\mathbb{T}^{\mathbb{G}}_{k,d}$, and write $\mathcal{H}_n \to_{\text{loc}} \mathbb{T}^{\mathbb{G}}_{k,d}$, if for all $t\ge 0$ and $i\in[\tau_v]$, $B_t(I_{n,i})$ converges to $\mathbb{T}^{\mathbb{G}}_{k,d}(t,i)$ in distribution with respect to the joint law $\mathbb{P}_n$ of $(\mathcal{H}_n, I_{n,i})$: that is,

$$\lim_{n\to\infty} \mathbb{P}_n\left(B_t(I_{n,i}) \cong \mathbb{T}^{\mathbb{G}}_{k,d}(t,i)\right)=1,$$

where $\cong$ denotes isomorphism which preserves vertex- and hyperedge-types and the incidence relation.

Consider the natural uniform random walk on the incidence graph of $\mathbb{T}^{\mathbb{G}}_{k,d}$, and its projection onto the finitely many disjoint orbits (types) for vertices and hyperedges, which gives a (bipartite) finite Markov chain. It is quite amazing to see that the reversibility of this projected chain determines whether there exists a sequence of finite hypergraphs that converge locally to $\mathbb{T}^{\mathbb{G}}_{k,d}$.

▶ **Theorem 27.** *Let $\mathbb{G}$ be an automorphism group of $\mathbb{T}_{k,d}$ with finitely many orbits for vertices and hyperedges. Let $\boldsymbol{D}$ and $\boldsymbol{K}$ be the branching matrices that corresponds to $\mathbb{G}$ as defined in Section 3. There is a sequence of random finite hypergraphs $\mathcal{H}_n \to_{\mathrm{loc}} \mathbb{T}^{\mathbb{G}}_{k,d}$ if and only if the Markov chain $\boldsymbol{P} = \begin{bmatrix} \boldsymbol{0} & \frac{1}{d+1}\boldsymbol{D} \\ \frac{1}{k+1}\boldsymbol{K} & \boldsymbol{0} \end{bmatrix}$ is time-reversible.*

We say a uniform random walk over a hypergraph $\mathcal{H}$ is a uniform random walk on the incidence graph of $\mathcal{H}$: that is, a random walk moves between vertices and hyperedges. Then the Markov chain $\boldsymbol{P}$ is the projection of the uniform random walk over $\mathbb{T}_{k,d}$ onto the equivalent classes of vertices and hyperedges (i.e. the orbits of the automorphism group $\mathbb{G}$ that corresponds to the $\boldsymbol{D}$ and $\boldsymbol{K}$). Meanwhile, matrix $\begin{bmatrix} \boldsymbol{0} & \boldsymbol{D} \\ \boldsymbol{K} & \boldsymbol{0} \end{bmatrix}$ is the adjacent matrix for a directed bipartite graph that describes the (weighted) incidence relation between vertex- and hyperedge-types in the following way: each directed bipartite edge from vertex-type-i to hyperedge-type-j (or vice versa) is assigned with weight d_{ij} (or k_{ji}). So the Markov chain $\boldsymbol{P}$ is also the random walk on this directed bipartite graph where the transition probability of each directed edge is proportional to its weight.

For the bipartite Markov chain $\boldsymbol{P}$, recall that due to Proposition 11, $\boldsymbol{P}$ must be irreducible. Then the time-reversibility of $\boldsymbol{P}$ is equivalent to the following: There exist positive vectors $\vec{p} = (p_i)_{i\in[\tau_v]}$ and $\vec{q} = (q_j)_{j\in[\tau_e]}$ that satisfy the bipartite detailed balanced equation:

$$p_i d_{ij} = q_j k_{ji}$$

for every $(i, j) \in [\tau_v] \times [\tau_e]$. Without loss of generality, we assume $\sum_i p_i + \sum_j q_j = 1$.

In fact, it is easy to check that $\vec{p}\boldsymbol{D} = (k+1)\vec{q}$ and $\vec{q}\boldsymbol{K} = (d+1)\vec{p}$, therefore the $\vec{p}$ and $\vec{q}$ are respectively the left eigenvector of $\boldsymbol{DK}$ and $\boldsymbol{KD}$ both with eigenvalue $(d+1)(k+1)$. Since both $\boldsymbol{DK}$ and $\boldsymbol{KD}$ are irreducible, due to the Perron-Frobenius theorem, the only positive left eigenvectors $\vec{p}$ and $\vec{q}$ are the ones that are associated with the Perron-Frobenius eigenvalue $(d+1)(k+1)$ and are one-dimensional.

Furthermore, it must holds that $\frac{\|\vec{p}\|_1}{\|\vec{q}\|_1} = \frac{k+1}{d+1}$. Denote $\vec{p'} = \frac{\vec{p}}{\|\vec{p}\|_1}$ and $\vec{q'} = \frac{\vec{q}}{\|\vec{q}\|_1}$. We have $\|\vec{p'}\| = \|\vec{q'}\| = 1$ and $p'_i \frac{d_{ij}}{d+1} = q'_j \frac{k_{ji}}{k+1}$ for every $(i, j) \in [\tau_v] \times [\tau_e]$, i.e. $\vec{p'}$ is the *vertex-stationary distribution* and $\vec{q'}$ is the *edge-stationary distribution*. We will mostly use $\vec{p}$ and $\vec{q}$ in our proof of Theorem 27. Recall for the automorphism group $\widehat{\mathbb{G}}$ defined in Section 3 such that $\lambda_c(\mathbb{T}^{\widehat{\mathbb{G}}}_{k,d}) = \lambda_c(\mathbb{T}_{k,d}) = \frac{d^d}{k(d-1)^{d+1}}$, i.e. the uniqueness of $\widehat{\mathbb{G}}$-translation-invariant Gibbs measures on $\mathbb{T}_{k,d}$ represents the uniqueness of all Gibbs measures on $\mathbb{T}_{k,d}$, the branching matrices are given as $\widehat{\boldsymbol{D}} = \begin{bmatrix} 1 & d \\ d & 1 \end{bmatrix}$ and $\widehat{\boldsymbol{K}} = \begin{bmatrix} k & 1 \\ 1 & k \end{bmatrix}$. It is easy to verify that the resulting Markov chain $\widehat{\boldsymbol{P}}$ is not time-reversible. It then follows from Theorem 27 that there does not exist *any* sequence of random finite hypergraphs that converge locally to $\mathbb{T}_{k,d}$ with the symmetry $\widehat{\mathbb{G}}$ assumed by the extremal Gibbs measures μ^+, μ^- whose uniqueness represents the uniqueness of all Gibbs measures.

▶ Remark 28. Given branching matrices $\boldsymbol{D}$ and $\boldsymbol{K}$, instead of considering $\mathcal{H}_n$ that converges locally for every type to the $\mathbb{T}^{\mathbb{G}}_{k,d}$ as in Definition 26, we can alternatively define a sequence $\mathcal{H}_n$ that *converges locally in average* to the $\mathbb{T}^{\mathbb{G}}_{k,d}$: that is, for all $t > 0$, the $B_t(I_n)$ converges to $\mathbb{T}^{\mathbb{G}}_{k,d}(t, I)$ in distribution, where I_n denotes a uniformly random vertex in the finite hypergraph $\mathcal{H}_n$, and I denotes a random vertex-type chosen according to the vertex-stationary distribution $\vec{p'}$. This definition looks more analogous to the local convergence defined in [31] for the anti-ferromagnetic 2-spin system. But we will see the two definitions are equivalent: A sequence $\mathcal{H}_n \to_{\mathrm{loc}} \mathbb{T}^{\mathbb{G}}_{k,d}$ also converges locally to $\mathbb{T}^{\mathbb{G}}_{k,d}$ in average, since by double counting the portion of vertices of type-i must converge to p'_i as $n \to \infty$; and conversely, a sequence converges locally to $\mathbb{T}^{\mathbb{G}}_{k,d}$ in average must also have$\mathcal{H}_n \to_{\text{loc}} \mathbb{T}^{\mathbb{G}}_{k,d}$, simply because neighborhoods of vertices of different types cannot be isomorphic to each other.

Proof of Theorem 27. We will prove the necessity of the reversibility of the chain by a double counting argument and the sufficiency is proved by explicitly constructing the sequence of the finite hypergraphs.

Double counting

Let $\mathcal{H}_n = (V_n, E_n)$ where $V_n = \biguplus_{s\in\tau_v} V_{n,s}$ and $E_n = \biguplus_{t\in\tau_e} E_{n,t}$. Assume that $\mathcal{H}_n \to_{\text{loc}} \mathbb{T}^{\mathbb{G}}_{k,d}$.

For $\mathbb{T}^{\mathbb{G}}_{k,d}$ such that there is a hypergraph sequence $\mathcal{H}_n = (V_n = \biguplus_{s\in\tau_v} V_{n,s},\ E_n = \biguplus_{t\in\tau_e} E_{n,t})$ converging locally to $\mathbb{T}^{\mathbb{G}}_{k,d}$, we show that the Markov chain $\boldsymbol{P}$ is time reversible. The proof is by a double counting of the number of vertex-hyperedge pairs with specific type combination.

Since the 1-neighborhood of the vertex with each type in $\mathcal{H}_n$ converges in distribution to the 1-neighborhood of the vertex with the same type in $\mathbb{T}^{\mathbb{G}}_{k,d}$, for sufficiently large n, we have all but a $o(1)$-fraction of vertices in $\mathcal{H}_n$ whose local transitions between vertex-types and hyperedge-types within 1-step are given precisely by $\boldsymbol{D}$ and $\boldsymbol{K}$. Thus, for every $(i,j) \in [\tau_v] \times [\tau_e]$, the total number of incident vertex-hyperedge pair (v,e) with $v \in V_{n,i}$ and $e \in E_{n,j}$ (counted from the vertex-side and from the hyperedge-side) is given by

$$d_{ij}(|V_{n,i}| + o(1)) = k_{ji}(|E_{n,j}| + o(1)).$$

As $n \to \infty$, we will have $(d_{ij}|V_{n,i}|)/(k_{ji}|E_{n,j}|) \to 1$, or equivalently

$$\frac{|E_{n,j}|}{|V_{n,i}|} \to \frac{d_{ij}}{k_{ji}}$$

for all $(i,j) \in [\tau_v] \times [\tau_e]$ such that $d_{ij}, k_{ji} \neq 0$. Thus there exists positive p_i, q_j such that $q_j/p_i = d_{ij}/k_{ji}$ for all such (i,j). Since $\boldsymbol{DK}$ and $\boldsymbol{KD}$ are irreducible, we have unique corresponding positive left eigenvectors, which is $(p_i)_{i\in[\tau_v]}, (q_j)_{j\in[\tau_e]}$ here, such that $p_i d_{ij} = q_j k_{ji}$ for all (i,j).

Construction of $\mathcal{H}_n$

Assume the Markov chain $\boldsymbol{P}$ in Theorem 27 to be time-reversible, and let $\vec{p} = (p_i)_{i\in[\tau_v]}$ and $\vec{q} = (q_j)_{j\in[\tau_e]}$ be the unique positive vectors satisfying $p_i d_{ij} = q_j k_{ji}$ for every $(i,j) \in [\tau_v] \times [\tau_e]$ and $\sum_i p_i + \sum_j q_j = 1$. The sequence of finite hypergraph sequence $\mathcal{H}_n$ that converges locally to $\mathbb{T}_{\boldsymbol{K},\boldsymbol{D}}$ is constructed as follows. The number n is approximately the total number of vertices and hyperedges in $\mathcal{H}_n$ (where the approximation is due to rounding).

- For each $s \in [\tau_v]$ and $t \in [\tau_e]$, let $V_{n,s}$ be the set of $\lceil p_s n \rceil$ vertices of type s, and $E_{n,t}$ be the set of $\lceil q_t n \rceil$ hyperedges of type t. We then describe hypergraphs $\mathcal{H}_n = (V_n, E_n)$ where $V_n = \biguplus_{s\in\tau_v} V_{n,s}$ and $E_n = \biguplus_{t\in\tau_e} E_{n,t}$.
- For each $s \in [\tau_v]$ and $t \in [\tau_e]$, let $N_{s,t} \triangleq \lceil d_{st} p_s n \rceil = \lceil k_{ts} q_t n \rceil$. Sample a uniformly random permutation $f : [N_{s,t}] \to [N_{s,t}]$, and create an incidence between the i-th vertex in $V_{n,s}$ and the j-th hyperedge in $E_{n,t}$ for every $(a, b = f(a))$ with $a \in i + \lceil p_s n \rceil \mathbb{Z}$ and $b \in j + \lceil q_t n \rceil \mathbb{Z}$.

Note that as normalized Perron eigenvectors for irreducible integer matrices, the $\vec{p}$ and $\vec{q}$ must be rational. Then there are infinitely many n such that $N_{s,t}/|V_{n,s}| = d_{st}$ and

$N_{s,t}/|E_{n,t}| = k_{ts}$. Without loss of generality, we can consider only these n, since otherwise it will contribute at most $o(1)$-fractions of bad neighborhoods.

Viewing multi-edges in the incidence graph of $\mathcal{H}_n$ as different edges, it holds that each vertex of type-s is incident to exactly d_{st} hyperedges of type-t and each hyperedge of type-t is incident by exactly k_{ts} vertices of type-s. Therefore it is sufficient to show that for any finite $r > 0$ the probability that the r-neighborhood of a vertex in $\mathcal{H}_n$ has no circle is 1 as $n \to \infty$, i.e. almost surely the r-neighborhood of a vertex in $\mathcal{H}_n$ is a hypertree. This can be proved easily by a standard routine of Galton-Watson branching process (see e.g. Ch. 9 in [15]) since the neighborhood is of constant size and the probability of reencountering a vertex or an edge from a population whose size goes to ∞ goes to 0. ◀

References

1 Mohsen Bayati, David Gamarnik, Dimitriy Katz, Chandra Nair, and Prasad Tetali. Simple deterministic approximation algorithms for counting matchings. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC)*, pages 122–127, 2007.

2 Ivona Bezakova, Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Daniel Stefankovic. Counting independent sets in hypergraphs when strong spatial mixing fails. *arXiv preprint arXiv:1510.09193*, 2015.

3 Magnus Bordewich, Martin Dyer, and Marek Karpinski. Path coupling using stopping times and counting independent sets and colorings in hypergraphs. *Random Structures & Algorithms*, 32(3):375–399, 2008.

4 Amir Dembo, Andrea Montanari, et al. Ising models on locally tree-like graphs. *The Annals of Applied Probability*, 20(2):565–592, 2010.

5 Andrzej Dudek, Marek Karpinski, Andrzej Ruciński, and Edyta Szymańska. Approximate counting of matchings in (3, 3)-hypergraphs. In *SWAT*, pages 380–391, 2014.

6 Martin Dyer, Leslie A Goldberg, and Mark Jerrum. Counting and sampling H-colourings. In *RANDOM*, pages 51–67, 2002.

7 Andreas Galanis, Qi Ge, Daniel Štefankovič, Eric Vigoda, and Linji Yang. Improved inapproximability results for counting independent sets in the hard-core model. *Random Structures & Algorithms*, 45(1):78–110, 2014.

8 Andreas Galanis and Leslie Ann Goldberg. The complexity of approximately counting in 2-spin systems on k-uniform bounded-degree hypergraphs. *arXiv preprint arXiv:1505.06146*, 2015.

9 Andreas Galanis, Daniel Stefankovic, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic Ising and hard-core models. arXiv preprint arXiv:1203.2226, 2012.

10 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability for antiferromagnetic spin systems in the tree non-uniqueness region. In *Proceedings of the 46th ACM Symposium on Theory of Computing (STOC)*, pages 823–831, 2014.

11 David Gamarnik and Dmitriy Katz. Correlation decay and deterministic FPTAS for counting colorings of a graph. *Journal of Discrete Algorithms*, 12:29–47, 2012.

12 David Gamarnik, Dmitriy Katz, and Sidhant Misra. Strong spatial mixing of list coloring of graphs. *Random Structures & Algorithms*, 2013.

13 Ole J Heilmann. Existence of phase transitions in certain lattice gases with repulsive potential. *Lettere Al Nuovo Cimento (1971–1985)*, 3(3):95–98, 1972.

14 Ole J Heilmann and Elliott H Lieb. Theory of monomer-dimer systems. *Communications in Mathematical Physics*, 25(3):190–232, 1972.

15 Svante Janson, Tomasz Luczak, and Andrzej Rucinski. *Random graphs*, volume 45. John Wiley & Sons, 2011.

16 Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18(6):1149–1178, 1989.

17 Marek Karpinski, Andrzej Rucinski, and Edyta Szymanska. Approximate counting of matchings in sparse uniform hypergraphs. In *Proceedings of the Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 72–79, 2013.

18 Frank P Kelly. Stochastic models of computer communication systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(3):379–395, 1985.

19 Liang Li, Pinyan Lu, and Yitong Yin. Approximate counting via correlation decay in spin systems. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 922–940, 2012.

20 Liang Li, Pinyan Lu, and Yitong Yin. Correlation decay up to uniqueness in spin systems. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 67–84, 2013.

21 Chengyu Lin, Jingcheng Liu, and Pinyan Lu. A simple FPTAS for counting edge covers. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 341–348, 2014.

22 Jingcheng Liu and Pinyan Lu. FPTAS for counting monotone CNF. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1531–1548, 2015.

23 Pinyan Lu, Menghui Wang, and Chihao Zhang. FPTAS for weighted fibonacci gates and its applications. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 787–799, 2014.

24 Pinyan Lu, Kuan Yang, and Chihao Zhang. Fptas for hardcore and ising models on hypergraphs. *arXiv preprint arXiv:1509.05494*, 2015.

25 Pinyan Lu and Yitong Yin. Improved FPTAS for multi-spin systems. In *RANDOM*, pages 639–654, 2013.

26 Elchanan Mossel, Dror Weitz, and Nicholas Wormald. On the hardness of sampling independent sets beyond the tree threshold. *Probability Theory and Related Fields*, 143(3-4):401–439, 2009.

27 Ricardo Restrepo, Jinwoo Shin, Prasad Tetali, Eric Vigoda, and Linji Yang. Improved mixing condition on the grid for counting and sampling independent sets. *Probability Theory and Related Fields*, 156(1-2):75–99, 2013.

28 Alistair Sinclair, Piyush Srivastava, and Marc Thurley. Approximation algorithms for two-state anti-ferromagnetic spin systems on bounded degree graphs. *Journal of Statistical Physics*, 155(4):666–686, 2014.

29 Alistair Sinclair, Piyush Srivastava, and Yitong Yin. Spatial mixing and approximation algorithms for graphs with bounded connective constant. In *Proceedings of the 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 300–309, 2013.

30 Allan Sly. Computational transition at the uniqueness threshold. In *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 287–296, 2010.

31 Allan Sly, Nike Sun, et al. Counting in two-spin models on d-regular graphs. *The Annals of Probability*, 42(6):2383–2416, 2014.

32 Frank Spitzer. Markov random fields on an infinite tree. *The Annals of Probability*, pages 387–398, 1975.

33 Dror Weitz. Combinatorial criteria for uniqueness of Gibbs measures. *Random Structures & Algorithms*, 27(4):445–475, 2005.

34 Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the 38th ACM Symposium on Theory of Computing (STOC)*, pages 140–149, 2006.

Sampling in Potts Model on Sparse Random Graphs[*]

Yitong Yin[†1] **and Chihao Zhang**[‡2]

1 **Nanjing University, Nanjing, China**
`yinyt@nju.edu.cn`

2 **Shanghai Jiao Tong University, Shanghai, China**
`chihao.zhang@gmail.com`

Abstract

We study the problem of sampling almost uniform proper q-colorings in sparse Erdős-Rényi random graphs $\mathcal{G}(n, d/n)$, a research initiated by Dyer, Flaxman, Frieze and Vigoda [2]. We obtain a fully polynomial time almost uniform sampler (FPAUS) for the problem provided $q > 3d + 4$, improving the current best bound $q > 5.5d$ [6].

Our sampling algorithm works for more generalized models and broader family of sparse graphs. It is an efficient sampler (in the same sense of FPAUS) for anti-ferromagnetic Potts model with activity $0 \le \beta < 1$ on $\mathcal{G}(n, d/n)$ provided $q > 3(1-\beta)d + 4$. We further identify a family of sparse graphs to which all these results can be extended. This family of graphs is characterized by the notion of contraction function, which is a new measure of the average degree in graphs.

1998 ACM Subject Classification F.2.2 Computations on Discrete Structures

Keywords and phrases Potts model, Sampling, Random Graph, Approximation Algorithm

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.47

1 Introduction

We study the problem of sampling almost uniform proper q-colorings in sparse Erdős-Rényi random graphs $\mathcal{G}(n, d/n)$. A classic sampling problem is to sample proper q-colorings of graphs with bounded *maximum degree* when $q \ge \alpha\Delta + \beta$, where Δ is the maximum degree. There is a substantial body of works on the problem [16, 1, 25, 3, 20, 14, 15, 13, 4, 10, 19]. The best positive result for this fundamental problem is the MCMC sampler for $q > \frac{11}{6}\Delta$ by Vigoda [25], and the best lower bound is due to Galanis, Štefankovič and Vigoda [9], which proved that the problem is intractable to solve when $q < \Delta$, even restricted to triangle-free Δ-regular graphs. The critical threshold $q = \Delta + 1$ is of great significance because it is the uniqueness threshold for the Δ-regular tree [18].

The studies of sampling proper q-colorings of graphs with bounded *average degree*, in particular the Erdős-Rényi random graph $\mathcal{G}(n, d/n)$ with constant d, was initiated in the seminal work of Dyer, Flaxman, Frieze and Vigoda [2], in which an algorithm was given to solve the problem with $q = \Theta(\log\log n/\log\log\log n)$ colors, substantially fewer than the maximum degree $\Theta(\log n/\log\log n)$ of the random graph. Several improvements have been

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

† Yitong Yin is supported in part by NSFC grants 61272081 and 61321491.

‡ Chihao Zhang is supported in part by NSFC grant 61373029.

done since then. A significant step was made by Efthymiou and Spirakis [8] and independently by Mossel and Sly [21], in which the bound of q was improved to a constant $f(d)$ which is a large enough polynomial of d. Most recently, in a breakthrough by Efthymiou [6], an efficient algorithm was given to solve the problem when $q > 5.5d$, in linear of average degree d.

In all aforementioned results, the algorithms are FPAUSes (fully polynomial-time almost uniform samplers), meaning that for any $\epsilon > 0$ the algorithms terminates in time polynomial in n and $1/\epsilon$ and returns a random proper q-coloring according to a distribution within total variation distance ϵ from the uniform distribution over all proper q-colorings of the graph. For a much weaker goal where the total variation distance ϵ is fixed, an elegant combinatorial algorithm was given by Efthymiou [5, 7] to solve the problem for all $q > d$, approaching the uniqueness threshold.

In this paper, we consider FPAUS for proper q-colorings of $\mathcal{G}(n, d/n)$ with constant d. We give an algorithm which achieves an improved bound $q > 3d + O(1)$.

▶ **Theorem 1.** *For all sufficiently large constant d, all finite $q > 3d + 4$, there is an FPAUS for proper q-colorings of $G \sim \mathcal{G}(n, d/n)$ whp.*

The result is established in a more general context, namely the anti-ferromagnetic Potts model. In the q-state Potts model with activity β, given a graph $G = (V, E)$, a *configuration* $\sigma \in [q]^V$ assigns each vertex $v \in V$ one of the q colors from $[q]$, and is assigned with the weight

$$w(\sigma) = \prod_{uv \in E} \beta^{\mathbf{1}(\sigma(u)=\sigma(v))}.$$

The *Gibbs distribution* over all configurations $\sigma \in [q]^V$, denoted by $\mu = \mu_{q,\beta,G}$, is defined as $\mu(\sigma) = w(\sigma)/Z$ where the normalizing factor $Z = \sum_\sigma w(\sigma)$ is the *partition function*. When $0 \le \beta < 1$, the model is *anti-ferromagnetic*, meaning that adjacent vertices favor disagreeing colors. In particular, when $\beta = 0$ the Gibbs distribution is the uniform distribution over all proper q-colorings of G. In [19], it was discovered that sampling from Potts model is tractable for any q when $3(1-\beta)\Delta < \beta$, and the lower bound in [9] shows that it is intractable to sample in the anti-ferromagnetic Potts model on triangle-free Δ-regular graph for any even q when $q < (1-\beta)\Delta$.

We give the following sampling algorithm for anti-ferromagnetic Potts model on sparse random graphs.

▶ **Theorem 2.** *For all sufficiently large constant d, all $0 \le \beta < 1$ and $q > 3(1-\beta)d+4$, there is an algorithm such that for $G \sim \mathcal{G}(n, d/n)$ and any $\epsilon > 0$, the algorithm terminates in time polynomial in n and $1/\epsilon$ and returns a random q-coloring of G, according to a distribution within total variation distance ϵ from the Potts Gibbs distribution $\mu_{q,\beta,G}$ whp (with respect to the law of $\mathcal{G}(n, d/n)$).*

In particular, when $\beta = 0$, the above algorithm is an FPAUS for proper q-colorings of $G \sim \mathcal{G}(n, d/n)$ for $q > 3d + 4$. Theorem 1 is a special case of Theorem 2.

Our algorithm on graphs with bounded average degree asymptotically approaches the lower bound in [9] in terms of maximum degree.

In fact, the algorithm in Theorem 2 works for any family of graphs characterized by a particular *contraction function*. We introduce the notion of contraction function to generalize the *connective constant* [24, 23], a notion of average degree extensively studied in statistical physics. Therefore, the algorithm stated in Theorem 2 does not only work for Erdős-Rényi random graph but also for families of sparse graphs with a proper notion of bounded average degree. In particular, it holds for graphs with bounded maximum degree Δ when $q > 3(1-\beta)\Delta + 1$, which also greatly improves the existing upper bounds for anti-ferromagnetic Potts model on graphs with bounded maximum degree [10, 19]. The definition of contraction function and the full statement of the main result with it are quite technical. We defer them to Section 2.

1.1 Techniques

In most of the previous works [2, 8, 21, 6], the sampling algorithms were based on block Glauber dynamics. For proper q-colorings, if the degree of a vertex is much higher than q, then the standard Glauber dynamics will have torpid mixing around that vertex since the color of that vertex will be frozen for most of the time. In previous works this was overcome by using block dynamics, such that within a block the high-degree vertices are hidden in the block's "core", which is separated from the block's boundary by an intermediate "buffer" of low-degree vertices. It is not hard to imagine that the construction of such blocks can be quite complicated and the efficient construction of blocks crucially relies on the sparsity of Erdős-Rényi random graph $\mathcal{G}(n, d/n)$.

In contrast, we use the correlation decay technique. This approach was introduced to multi-spin models (e.g. colorings) in the seminal work of Gamarnik and Katz [10], in which they gave an FPAUS for proper q-colorings when $q > 2.844\Delta$ where Δ is the maximum degree. This was later improved to $q > 2.581\Delta$ in [19], which remains to be the best bound achieved by correlation-decay-based algorithms for proper q-colorings.

Our algorithm heavily relies on the computation tree recursion introduced in [10]. The basic idea is simple: sampling with the estimations of marginal probabilities, which are computed approximately by a proper truncation of the the computation tree recursion. With correlation decay, the approximation is accurate enough so the algorithm is an FPAUS. A complication here is that the degrees of vertices are unbounded. We overcome this by introducing a computation tree in terms of blocks and establish the decay of correlation between blocks.

The blocks in our algorithm can be constructed straightforwardly: they are just clusters of high-degree vertices. Due to the simple and generic construction of blocks, our algorithm may work for general families of graphs, and can be applied as a generic method for graphs with a few high-degree vertices.

The idea of block correlation decay was introduced in our previous work [26] to establish the correlation decay for proper q-colorings of $\mathcal{G}(n, d/n)$ for $q > 2d + O(1)$, by a block modification to another recursion of Gamarnik, Katz and Misra [11]. This recursion is suitable for proving "correlation decay only" result. A drawback of the current approaches based on correlation decay is that we do not know how to use this approach to get an algorithm achieving a bound which is close to $q > 2\Delta + O(1)$, even on graphs with maximum degree Δ. Sampling proper q-coloring in $\mathcal{G}(n, d/n)$ for $q > 2d$ or smaller q may require new understandings of correlation decay in multi-spin systems, or may have to use other techniques such as Glauber dynamics.

2 Preliminary and statement of the main result

Let $G = (V, E)$ be an undirected graph. For any subset $S \subseteq V$ of vertices, let $G[S]$ denote the subgraph of G induced by S, and let $\partial B = \{u \in V \setminus B \mid \exists w \in B, (u, w) \in E\}$ denote the vertex boundary of B. Given a vertex v in G, let $\mathrm{dist}_G(v, S)$ denote the minimum distance from v to any vertex $u \in S$ in G. In case that $S = \{u\}$ is a singleton, we write $\mathrm{dist}_G(v, u)$ instead of $\mathrm{dist}_G(v, S)$.

Potts model

The anti-ferromagnetic Potts model is parameterized by an integer $q \geq 2$ and an *activity* parameter $0 \leq \beta < 1$. Each element of $[q]$ represents a *color* or a *state*. Let $G = (V, E)$ be a graph. A *configuration* $\sigma \in [q]^\Lambda$ on a subset $\Lambda \subseteq V$ of vertices assigns each vertex v in Λ one of the q colors in $[q]$. In the Potts model on graph G, each configuration $\sigma \in [q]^V$ is assigned a weight

$$w_G(\sigma) = \beta^{\#\mathsf{mon}(\sigma)},$$

where $\#\mathsf{mon}(\sigma) = |\{(u, v) \in E \mid \sigma(u) = \sigma(v)\}|$ gives the number of monochromatic (undirected) edges in the configuration σ.

The analysis of correlation decay introduces Potts model with boundary conditions. More formally, we consider an instance of Potts model as a tuple $\Omega = (G, \Lambda, \sigma)$ where $G = (V, E)$ is an undirected graph, $\Lambda \subseteq V$ is a subset of vertices in G and $\sigma \in [q]^\Lambda$ is a configuration on Λ. Given such an instance $\Omega = (G, \Lambda, \sigma)$, the weight function w_Ω assigns each configuration $\pi \in [q]^V$ the weight $w_\Omega(\pi) = w_G(\pi)$ if π agrees with σ over all vertices in Λ, and $w_\Omega(\pi) = 0$ if otherwise. An instance Ω is *feasible* if there exists a configuration on V with positive weight. This gives rise to a natural probability distribution $\mu = \mu_{q,\beta,G}$, called *Gibbs distribution*, over all configurations $\pi \in [q]^V$ for a feasible Potts instance:

$$\mu(\pi) = \mathbf{Pr}_\Omega[c(V) = \pi] = \frac{w_\Omega(\pi)}{Z(\Omega)},$$

where $Z(\Omega) = \sum_{\sigma \in [q]^V} w_\Omega(\sigma)$ is the *partition function*. For a vertex $v \in V$ and any color $x \in [q]$, we use $\mathbf{Pr}_\Omega[c(v) = x]$ to denote the marginal probability that v is assigned color x by a configuration sampled from the Gibbs distribution. Similarly, for a set $S \subseteq V$ and $\pi \in [q]^S$, we use $\mathbf{Pr}_\Omega[c(S) = \pi]$ to denote the marginal probability that S is assigned configuration π by a configuration sampled from the Gibbs distribution.

Block and sparsity

Fix any $q \geq 2$ and $0 \leq \beta < 1$. Let $\Omega = (G, \Lambda, \sigma)$ be an instance of q-state Potts model with activity β and v a vertex in G. We call v a *low-degree* vertex if $\deg_G(u) < \frac{q-1}{1-\beta} - 2$, and otherwise we call it a *high-degree* vertex.

▶ **Definition 3** (permissive block)**.** Let $\Omega = (G, \Lambda, \sigma)$ be a Potts instance where $G = (V, E)$. A vertex set $B \subseteq V \setminus \Lambda$ is a *permissive block* in Ω if every boundary vertex $u \in \partial B \setminus \Lambda$ is a low-degree vertex. For any subset of vertices $S \subseteq V \setminus \Lambda$, we denote $B(S) = B_\Omega(S)$ the *minimal permissive block* containing S. We write $B(v) = B(S)$ if $S = \{v\}$ is a singleton.

▶ **Definition 4.** A family $\mathcal{G}$ of finite graphs is *locally sparse* if there exists a constant $C > 0$ such that for every $G = (V, E)$ in the family and every path P in G of length ℓ we have $|B(P)| \leq C(\ell + \log |V|)$.

SAW tree

Given a graph $G = (V, E)$ and a vertex $v \in V$, a rooted tree T can be naturally constructed from all self-avoiding walks starting from v in G as follows: Each vertex in T corresponds to a self-avoiding walk (simple path in G) $P = (v, v_1, v_2, \dots, v_k)$ starting from v, whose children correspond to all self-avoiding walks $(v, v_1, v_2, \dots, v_k, v_{k+1})$ in G extending P, and the root of T corresponds to the trivial walk (v). The resulting tree, denoted by $T_{\text{SAW}}(G, v)$, is called the *self-avoiding walk* (SAW) tree constructed from vertex v in graph G.

From this construction, every vertex in $T_{\text{SAW}}(G, v)$ can be naturally identified with the vertex in V (many-to-one) at which the corresponding self-avoiding walk ends.

Contraction function

Given a vertex v in a locally finite graph $G = (V, E)$, let $\mathsf{SAW}(v, \ell)$ denote the set of self-avoiding walks in G of length ℓ starting at v. The following notion of connective constant of families of finite graphs is introduced in [24].

▶ **Definition 5** (connective constant [24, 23])**.** Let $\mathcal{G}$ be a family of finite graphs. The *connective constant* of $\mathcal{G}$ is bounded by Δ if there exists a positive constant $C > 0$ such that for any graph $G = (V, E)$ in $\mathcal{G}$ and any vertex v in G, we have $|\mathsf{SAW}(v, \ell)| \le n^C \Delta^\ell$ where $n = |V|$ for all $\ell \ge 1$.

Let $\delta : \mathbb{N} \to \mathbb{R}^+$ be a function. Given a vertex v in a locally finite graph $G = (V, E)$, let

$$\mathcal{E}_\delta(v, \ell) := \sum_{\substack{(v, v_i, \dots, v_\ell) \\ \in \mathsf{SAW}(v, \ell)}} \prod_{i=1}^{\ell} \delta(\deg(v_i)). \tag{1}$$

▶ **Definition 6** (contraction function)**.** Let $\mathcal{G}$ be a family of finite graphs. The $\delta : \mathbb{N} \to \mathbb{R}^+$ is a *contraction function* for $\mathcal{G}$ if there exist positive constants $C > 0, \gamma < 1$ such that for any graph $G = (V, E)$ in $\mathcal{G}$ and any vertex v in G, we have $\mathcal{E}_\delta(v, \ell) < n^C \gamma^\ell$ where $n = |V|$ for all $\ell \ge 1$.

It is easy to see that graph families $\mathcal{G}$ with constant contraction function $\delta(d) = \frac{1}{\Delta}$ are precisely the families $\mathcal{G}$ of connective constant bounded strictly by Δ.

Statement of the main result

Now we are ready to state our main technical result.

▶ **Theorem 7** (Main theorem)**.** *Let $q \ge 3$ be an integer and $0 \le \beta < 1$. Let $\mathcal{G}$ be a family of finite graphs that satisfies the followings:*

- *the following $\delta(\cdot)$ is a contraction function for $\mathcal{G}$:*

$$\delta(d) = \begin{cases} \frac{2(1-\beta)}{q-1-(1-\beta)d} & \text{if } d \le \frac{q-1}{1-\beta} - 2, \\ 1 & \text{otherwise;} \end{cases} \tag{2}$$

- *$\mathcal{G}$ is locally sparse;*
- *(proper q-coloring) if $\beta = 0$, then $\mathcal{G}$ also needs to be q-colorable.*

Then there is an FPTAS for computing the partition function $Z(\Omega)$ for every $\Omega = (G, \Lambda, \sigma)$ with $G \in \mathcal{G}$. Consequently, there is an algorithm such that for all $G = (V, E) \in \mathcal{G}$, all $\epsilon > 0$, the algorithm terminates in time polynomial in $n = |V|$ and $1/\epsilon$, and returns a random $\sigma \in [q]^V$ according a distribution within total variation distance ϵ from the Potts Gibbs distribution $\mu_{q,\beta,G}$.

3 The computation tree for blocks

In this section, we introduce recursions to compute the marginal probabilities on a vertex and on a permissive block respectively.

When $\beta = 0$, the model becomes proper q-coloring, and the feasibility of a configuration becomes an issue.

Let $\Omega = (G, \Lambda, \sigma)$, where $G = (V, E)$ be a feasible instance of proper q-coloring. Recall that an instance $\Omega = (G, \Lambda, \sigma)$ is feasible if there exists a proper q-coloring consistent with σ. For a subset of vertices $S \subseteq V \setminus \Lambda$, a q-coloring $\pi \in [q]^S$ is (globally) *feasible* if it can be extended to a proper q-coloring of G. A q-coloring $\pi \in [q]^S$ is *locally feasible*, if $\sigma \cup \pi$ is a proper q-coloring in the subgraph $G[\Lambda \cup S]$ induced by $\Lambda \cup S$.

▶ **Proposition 8.** *Let $\Omega = (G, \Lambda, \sigma)$ where $G = (V, E)$ be a feasible instance of proper q-coloring, $v \in V \setminus \Lambda$ be a vertex and $\pi \in [q]^{B(v)}$ be a locally feasible configuration. Then π is also feasible.*

Proof. Denote $B = B(v)$. Fix a configuration $\eta \in [q]^V$ such that $w_\Omega(\eta) > 0$, this is possible since Ω is feasible. We denote by η' the restriction of η to $V \setminus ((B \cup \partial B) \setminus \Lambda)$, i.e., the set of vertices that are either in Λ, or not in $B \cup \partial B$.

Consider the configuration $\eta = \pi \cup \eta' \in [q]^{V \setminus (\partial B \setminus \Lambda)}$, it can be extended to a configuration $\rho \in [q]^V$ with $w_\Omega(\rho) > 0$ in a greedy fashion, since every vertex in $\partial B \setminus \Lambda$ is of low-degree. Thus ρ witness that π is feasible. ◀

With this proposition, we do not distinguish between local feasibility and feasibility of configurations on permissive blocks. For a permissive block B, we use $\mathcal{F}(B)$ to denote the set of feasible configuration. Note that when $\beta > 0$, the set $\mathcal{F}(B)$ is simply $[q]^B$.

3.1 The recursion

Let $\Omega = (G, \Lambda, \sigma)$ where $G = (V, E)$ be an instance of Potts model and $v \in V \setminus \Lambda$ be a vertex. Let $B = B(v)$ be the minimal permissive block containing v. Let $\delta B = \{u_i v_i \mid i \in [m]\}$ be an enumeration of boundary edges of B where $v_i \notin B$ for every $i \in [m]$. In this notation, more than one u_i or v_i may refer to the same vertex. We denote $E(B) := \{uv \in E \mid u, v \in B\}$ the edges in B. We use $\bar{B}$ to denote the inner boundary of B, i.e., $\bar{B} = \{u \in B \mid uv \in E \text{ and } v \notin B\}$.

Recall that we use $\mathcal{F}(B)$ to denote the set of feasible configurations on a permissive block B, it is easy to see that, for every $x \in [q]$,

$$\mathbf{Pr}_\Omega [c(v) = x] = \sum_{\substack{\pi \in \mathcal{F}(B): \\ \pi(v) = x}} \mathbf{Pr}_\Omega [c(B) = \pi] .$$

This identity relates the marginal probability on a vertex to marginal probabilities on a block. We now define notations for some sub-instances and give a block-to-vertices identity.

Let $\pi \in \mathcal{F}(B)$ be a configuration on a permissive block B. For every $i \in [m]$, denote $\pi_i = \pi(u_i)$. Let $G_B = (V_B, E_B)$ denote the graph obtained from G by removing $B \setminus \bar{B}$ and edges in $E(B)$, i.e., $V' = (V \setminus B) \cup \bar{B}$, $E' = E \setminus E(B)$. Let $\Omega_B = (G_B, \Lambda, \sigma)$. For every $i = 1, 2, \ldots, m+1$, define $\Omega_i^\pi = (G_i^\pi, \Lambda_i^\pi, \sigma_i^\pi)$ as the instance obtained from Ω_B by fixing u_j to color π_j for every $j \in [i-1]$ and by removing edges $u_j v_j$ for every $j = i, i+1, \ldots, m$.

▶ **Lemma 9.** *Assuming above notations, it holds that*

$$\mathbf{Pr}_{\Omega}\left[c(B)=\pi\right]=\frac{w_{G[B]}(\pi)\cdot\prod_{i=1}^{m}\left(1-(1-\beta)\mathbf{Pr}_{\Omega_i^{\pi}}\left[c(v_i)=\pi_i\right]\right)}{\sum_{\rho\in\mathcal{F}(B)}w_{G[B]}(\rho)\cdot\prod_{i=1}^{m}\left(1-(1-\beta)\mathbf{Pr}_{\Omega_i^{\rho}}\left[c(v_i)=\rho_i\right]\right)}. \tag{3}$$

Proof.

$$\begin{aligned}\mathbf{Pr}_{\Omega}\left[c(B)=\pi\right]&=\frac{w_{G[B]}(\pi)\cdot Z(\Omega_{m+1}^{\pi})}{\sum_{\rho\in\mathcal{F}(B)}w_{G[B]}(\rho)\cdot Z(\Omega_{m+1}^{\rho})}=\frac{w_{G[B]}(\pi)\cdot\frac{Z(\Omega_{m+1}^{\pi})}{Z(\Omega_1^{\pi})}}{\sum_{\rho\in\mathcal{F}(B)}w_{G[B]}(\rho)\cdot\frac{Z(\Omega_{m+1}^{\rho})}{Z(\Omega_1^{\rho})}}\\&=\frac{w_{G[B]}(\pi)\cdot\prod_{i=1}^{m}\frac{Z(\Omega_{i+1}^{\pi})}{Z(\Omega_i^{\pi})}}{\sum_{\rho\in\mathcal{F}(B)}w_{G[B]}(\rho)\cdot\prod_{i=1}^{m}\frac{Z(\Omega_{i+1}^{\rho})}{Z(\Omega_i^{\rho})}}.\end{aligned}$$

Since for every $\rho\in\mathcal{F}(B)$ and $i\in[d]$,

$$Z(\Omega_{i+1}^{\rho})=\sum_{y\in[q]}Z\left(\Omega_i^{\rho}\mid c(v_i)=y\right)\cdot\beta^{\mathbf{1}(y=\rho(u_i))}$$

where $Z\left(\Omega_i^{\rho}\mid c(v_i)=y\right)$ stands for the sum of the weights of all feasible configurations σ on Ω_i^{ρ} satisfying $\sigma(v_i)=y$ and $\mathbf{1}(\cdot)$ is the indicator function. With this identity, we can further write

$$\begin{aligned}\mathbf{Pr}_{\Omega}\left[c(B)=\pi\right]&=\frac{w_{G[B]}(\pi)\cdot\prod_{i=1}^{m}\frac{\sum_{y\in[q]}Z(\Omega_i^{\pi}\mid c(v_i)=y)\cdot\beta^{\mathbf{1}(y=\pi(u_i))}}{Z(\Omega_i^{\pi})}}{\sum_{\rho\in\mathcal{F}(B)}w_{G[B]}(\rho)\cdot\prod_{i=1}^{m}\frac{\sum_{y\in[q]}Z(\Omega_i^{\rho}\mid c(v_i)=y)\cdot\beta^{\mathbf{1}(y=\rho(u_i))}}{Z(\Omega_i^{\rho})}}\\&=\frac{w_{G[B]}(\pi)\cdot\prod_{i=1}^{m}\left(1-(1-\beta)\mathbf{Pr}_{\Omega_i^{\pi}}\left[c(v_i)=\pi_i\right]\right)}{\sum_{\rho\in\mathcal{F}(B)}w_{G[B]}(\rho)\cdot\prod_{i=1}^{m}\left(1-(1-\beta)\mathbf{Pr}_{\Omega_i^{\rho}}\left[c(v_i)=\rho_i\right]\right)}.\end{aligned}$$ ◀

This identity expresses the marginal probability on a permissive block as the function of marginal probabilities on its incident vertices, with modified instances. We now analyze the derivatives of this function, which is important in the analysis of correlation decay.

▶ **Lemma 10.** *Let* $\mathbf{p}=(p_{i,\rho})_{i\in[m],\rho\in\mathcal{F}(B)}$, $\hat{\mathbf{p}}=(\hat{p}_{i,\rho})_{i\in[m],\rho\in\mathcal{F}(B)}$ *be two tuples of variables and*

$$f(\mathbf{p}):=\frac{w_{G[B]}(\pi)\prod_{i=1}^{m}(1-(1-\beta)p_{i,\pi})}{\sum_{\rho\in\mathcal{F}(B)}w_{G[B]}(\rho)\prod_{i=1}^{m}(1-(1-\beta)p_{i,\rho})}.$$

Assume for every $i\in[m],\rho\in\mathcal{F}(B(v))$, $p_{i,\rho},\hat{p}_{i,\rho}\le\frac{1-\beta}{q-(1-\beta)d_i}$, *then*

$$\left|\log f(\mathbf{p})-\log f(\hat{\mathbf{p}})\right|\le\sum_{i\in[d]}\frac{2(1-\beta)}{q-(1-\beta)d_i-1}\cdot\max_{\rho\in\mathcal{F}(B(v))}\left|\log p_{i,\rho}-\log\hat{p}_{i,\rho}\right|.$$

Proof. For every $i\in[m]$, we have

$$\frac{\partial f}{\partial p_{i,\pi}}=-(1-\beta)f(1-f)\cdot\frac{1}{1-(1-\beta)p_{i,\pi}}.$$

For every $i\in[m]$ and $\rho\neq\pi$, we have

$$\frac{\partial f}{\partial p_{i,\rho}}=(1-\beta)f\cdot\frac{w_{G[B]}(\rho)\prod_{i=1}^{m}(1-(1-\beta)p_{i,\rho})}{\sum_{\sigma\in\mathcal{F}(B)}w_{G[B]}(\sigma)\prod_{i=1}^{m}(1-(1-\beta)p_{i,\sigma})}\cdot\frac{1}{1-(1-\beta)p_{i,\rho}}.$$

Thus,

$$\sum_{\substack{\rho\in\mathcal{F}(B)\\ \rho\neq\pi}} \frac{\partial f}{\partial p_{i,\rho}} \leq (1-\beta)f(1-f)\cdot \max_{\substack{\rho\in\mathcal{F}(B)\\ \rho\neq\pi}} \frac{1}{1-(1-\beta)p_{i,\rho}}.$$

Let $\Phi = \frac{1}{x}$, by mean value theorem, for some $\tilde{\mathbf{p}} = (\tilde{p}_{i,\rho})_{i\in[m],\rho\in\mathcal{F}(B(v))}$ where each $\tilde{p}_{i,\rho} \leq \frac{1-\beta}{q-(1-\beta)d_i}$, we have

$$\begin{aligned}
&|\log f(\mathbf{p}) - \log f(\hat{\mathbf{p}})| \\
&= \sum_{i\in[m]}\sum_{\rho\in\mathcal{F}(B)} \left(\frac{\Phi(f)}{\Phi(p_{i,\rho})}\left|\frac{\partial f}{\partial p_{i,\rho}}\right|\right)\Bigg|_{\mathbf{p}=\tilde{\mathbf{p}}} \cdot |\log p_{i,\rho} - \log \hat{p}_{i,\rho}| \\
&\leq \sum_{i\in[m]} \left(\frac{\Phi(f)}{\Phi(p_{i,\pi})}\left|\frac{\partial f}{\partial p_{i,\pi}}\right| + \sum_{\substack{\rho\in\mathcal{F}(B)\\ \rho\neq\pi}} \frac{\Phi(f)}{\Phi(p_{i,\rho})}\left|\frac{\partial f}{\partial p_{i,\rho}}\right|\right)\Bigg|_{\mathbf{p}=\tilde{\mathbf{p}}} \cdot \max_{\rho\in\mathcal{F}(B(v))} |\log p_{i,\rho} - \log \hat{p}_{i,\rho}| \\
&\leq \sum_{i\in[m]} \left((1-\beta)\left(\frac{p_{i,\pi}}{1-(1-\beta)p_{i,\pi}} + \max_{\substack{\rho\in\mathcal{F}(B)\\ \rho\neq\pi}} \frac{p_{i,\rho}}{1-(1-\beta)p_{i,\rho}}\right)\right)\Bigg|_{\mathbf{p}=\tilde{\mathbf{p}}} \\
&\qquad\qquad \cdot \max_{\rho\in\mathcal{F}(B(v))} |\log p_{i,\rho} - \log \hat{p}_{i,\rho}| \\
&\leq \sum_{i\in[m]} \frac{2(1-\beta)}{q-(1-\beta)d_i-(1-\beta)} \cdot \max_{\rho\in\mathcal{F}(B(v))} |\log p_{i,\rho} - \log \hat{p}_{i,\rho}|.
\end{aligned}$$ ◀

3.2 Bounds for marginals

The following lemma gives an upper bound for the probability $\mathbf{Pr}_\Omega[c(v)=x]$.

▶ **Lemma 11.** *Assume $q > (1-\beta)d$. For every color $x\in[q]$, it holds that*

$$\mathbf{Pr}_\Omega[c(v)=x] \leq \frac{1}{q-(1-\beta)d},$$

where d is the degree of v in G.

Proof. Assume $x=1$. For every $i\in[q]$, let x_i denote the number of neighbors of v that are of color i. Then $p_{v,1} \leq \max \frac{\beta^{x_1}}{\sum_{i\in[q]}\beta^{x_i}}$ subject to the constraints that all x_i are nonnegative integers and $\sum_{i=1}^q x_i = d$. Since $\beta\leq 1$, we can assume $x_1=0$, thus $p_{v,1}\leq \max\frac{1}{1+\sum_{i=2}^q\beta^{x_i}}$. We now distinguish between two cases:

1. (If $d\geq q-1$) In this case, let $\lambda = 1-\beta$, then

$$\frac{1}{1+\sum_{i=2}^q\beta^{x_i}} \leq \frac{1}{1+(q-1)(1-\lambda)^{\frac{d}{q-1}}} \overset{\heartsuit}{\leq} \frac{1}{1+(q-1)\left(1-\frac{\lambda d}{q-1}\right)} = \frac{1}{q-(1-\beta)d},$$

where $\heartsuit$ is due to the fact that the inequality $(1-a)^b \geq 1-ab$ holds when $0\leq a\leq 1$ and $b\geq 1$.

2. (If $d<q-1$) In this case, due to the integral constraint of x_i's, the term $\sum_{i=2}^q\beta^{x_i}$ minimizes when d of x_i's are set to one and remaining x_i's are set to zero. Therefore, we have

$$\frac{1}{1+\sum_{i=2}^q\beta^{x_i}} \leq \frac{1}{1+d\beta+(q-1-d)} = \frac{1}{q-(1-\beta)d},$$ ◀

Algorithm 1: $\texttt{marg}(\Omega, v, x, \ell)$

```
1 If $v$ is fixed to be color $y$, then return 1 if $x = y$ and return 0 if $x \neq y$;
2 If $\ell < 0$ return $1/q$;
3 Compute $B(v)$;
4 For every $\rho \in \mathcal{F}(B(v))$, let $\hat{p}_\rho \leftarrow \texttt{marg-block}(\Omega, B(v), \rho, \ell)$;
5 Return $\min\left\{ \sum_{\substack{\pi\in\mathcal{F}(B(v))\\ s.t.\ \pi(v)=x}} \hat{p}_\pi, \frac{1}{\max\{1, q-(1-\beta)\deg_G(v)\}} \right\}$
```

Algorithm 2: $\texttt{marg-block}(\Omega, B(v), \pi, \ell)$

```
1 Compute $P_i$ for every $i \in [m]$;
2 $\hat{p}_{i,\rho} \leftarrow \texttt{marg}(\Omega_i^\rho, v_i, \rho_i, \ell - |P_i|)$ for every $i \in [m]$ and $\rho \in \mathcal{F}(B)$;
3 Return $\frac{w_{G[B]}(\pi) \prod_{i\in[m]}(1-(1-\beta)\hat{p}_{i,\pi})}{\sum_{\rho\in\mathcal{F}(B)} w_{G[B]}(\rho) \prod_{i\in[m]}(1-(1-\beta)\hat{p}_{i,\rho})}$;
```

The recursion (3) holds for arbitrary set of vertices B (not necessary a permissive block), thus if one takes B as a single vertex, it implies the following simple lower bound for marginal probabilities on a vertex.

▶ **Lemma 12.** *For every feasible $x \in [q]$, it holds that*

$$\mathbf{Pr}_\Omega\left[c(v) = x\right] \geq \frac{\beta^d}{q},$$

where d is the degree of v in G.

3.3 The algorithm

We now implement the recursions introduced in previous sections to estimate marginals. There is a slight difference between the case of $\beta > 0$ and the case of $\beta = 0$. If $\beta = 0$, our algorithm may encounter an infeasible instance and we need to check the feasibility in advance.

The $\beta > 0$ case

We define two procedures $\texttt{marg}(\Omega, v, x, \ell)$ and $\texttt{marg-block}(\Omega, B(v), \pi, \ell)$ calling each other to estimate vertex and block marginal respectively. We assume $\Omega = (G, \Lambda, \sigma)$ with $G = (V, E)$ is an instance of Potts model, $v \in V \setminus \Lambda$ is a vertex, $x \in [q]$ is a color and ℓ is an integer. Recall that for a permissive block $B(v)$, we use $\mathcal{F}(B)$ to denote the set of feasible configurations over $B(v)$.

To describe the algorithm for estimating the block marginals, we need to introduce some notations. Let $B = B(v)$, and we enumerate the boundary edges in δB by $e_i = u_i v_i$ for $i = 1, 2, \ldots, m$, where $v_i \notin B$. With this notation more than one u_i or v_i may refer to the same vertex, which is fine. For every $i \in [m]$ and $\rho \in \mathcal{F}(B)$, define Ω_B and Ω_i^ρ as in Lemma 9.

Let $P_i = (v, w_1, w_2, \ldots, w_k, v_i)$ be a self-avoiding walk from v to v_i such that all intermediate vertices w_i are in $B(v)$. Since $B(v)$ is a minimal permissive block, such walk always exists, and let P_i be an arbitrary one of them if there are multiple ones.

```
Algorithm 3: marg($\Omega, v, x, \ell$)
1 If $v$ is fixed to be color $y$, then return 1 if $x = y$ and return 0 if $x \neq y$;
2 Compute $B(v)$;
3 If $\ell < 0$, then return $1/q$ if there is a feasible $\pi \in \mathcal{F}(B(v))$ such that $\pi(v) = x$ and
   return 0 if no such $\pi$ exists;
4 For every $\rho \in \mathcal{F}(B(v))$, let $\hat{p}_\rho \leftarrow$ marg-block($\Omega, B(v), \rho, \ell$);
5 Return min $\left\{ \sum_{\substack{\pi \in \mathcal{F}(B(v)) \\ s.t.\ \pi(v)=x}} \hat{p}_\pi, \frac{1}{\max\{1, q-(1-\beta)\deg_G(v)\}} \right\}$
```

The $\beta = 0$ case

We slightly modify our procedure to deal with infeasible instance. Let $\Omega = (G, \Lambda, \sigma)$ be an instance of Potts model with $q \geq 3$ and activity $\beta = 0$ where $G = (V, E)$, $v \in V \setminus \Lambda$ be a vertex, $x \in [q]$ be a color and ℓ be an integer. We define

The only difference of this version of `marg` is at step 3, where we check whether the color x is locally feasible. We return $1/q$ if so and return 0 otherwise.

4 Correlation decay

In this section, we show that the algorithms introduced in Section 3 to estimate marginals are accurate, if the input instance satisfies the conditions specified in the statement of Theorem 7.

▶ **Lemma 13.** *Let $q \geq 3$ be an integer and $0 \leq \beta < 1$ be a real. Let $\mathcal{G}$ be a family of finite graphs satisfying the conditions of Theorem 7.*

There exists an algorithm such that for every feasible instance $\Omega = (G, \Lambda, \sigma)$ of Potts model where $G = (V, E) \in \mathcal{G}$ with $|V| = n$, $\Lambda \subseteq V$ and $\sigma \in [q]^\Lambda$, for every vertex $v \in V$ and every color $x \in [q]$, it can compute an estimation $\hat{p}$ of $\mathbf{Pr}_\Omega[c(v) = x]$ in time polynomial in n, such that

$$1 - O\left(\frac{1}{n^3}\right) \leq \frac{\hat{p}}{\mathbf{Pr}_\Omega[c(v) = x]} \leq 1 + O\left(\frac{1}{n^3}\right).$$

To prove Lemma 13, we introduce the notion of error function to relate contraction function and the accurate of our estimation algorithm.

▶ **Definition 14.** Given an instance $\Omega = (G, \Lambda, \sigma)$ of Potts model with $q \geq 3$ and activity $0 \leq \beta < 1$ where $G = (V, E)$ with $|V| = n$. Let $v \in V \setminus \Lambda$ be a vertex, $T = T_{\mathrm{SAW}}(G[V], v)$ be the self-avoiding walk tree rooted at v in G and S be a set of vertices in T. Assume v has m children $v_1, v_2, \ldots, v_m$ in T, let T_i denote the subtree of T rooted at v_i. We recursively define the error function:

- Case $\beta > 0$:

$$\mathcal{E}_{T,S} := \begin{cases} \sum_{i=1}^m \delta(\deg_G(v_i)) \cdot \mathcal{E}_{T_i,S} & \text{if } v \notin S \cup \Lambda, \\ q + n \log \frac{1}{\beta} & \text{if } v \in S, \\ 0 & \text{if } v \in \Lambda. \end{cases}$$

- Case $\beta = 0$:

$$\mathcal{E}_{T,S} := \begin{cases} \sum_{i=1}^m \delta(\deg_G(v_i)) \cdot \mathcal{E}_{T_i,S} & \text{if } v \notin S \cup \Lambda, \\ n \log q & \text{if } v \in S, \\ 0 & \text{if } v \in \Lambda. \end{cases}$$

In the above definition, the set S specifies the boundary of our recursively defined error function $\mathcal{E}_{T,S}$. The error function $\mathcal{E}_{T,S}$ will be used as an upper bound for the error in our estimation algorithm. If the function $\delta(\cdot)$ is a contraction for a family $\mathcal{G}$, then for every graph $G = (V, E) \in \mathcal{G}$ and vertex $v \in V$, as we shall show in the next lemma, there exists a set of low-degree vertices in $T_{\mathrm{SAW}}(G, v)$ at certain depth. This set of vertices, as it will become clear later, serves as the boundary S in our computation tree.

▶ **Lemma 15.** *Let $\mathcal{G}$ be a family of finite graphs for which $\delta(\cdot)$ is a contraction function. Then for some constants $\theta > 1$ and $C > 0$, for every $G = (V, E) \in \mathcal{G}$ with $|V| = n$, every $v \in V$ and every $L \geq C \log n$, there exists a low-degree S in $T = T_{\mathrm{SAW}}(G, v)$ such that for every $u \in S$, $L < \mathrm{dist}_T(u, v) \leq \theta L$ and every self-avoiding walk in T from v of length θL intersects S.*

Proof. Let $G = (V, E) \in \mathcal{G}$ be a graph. It follows from the definition of contraction function that for some constant $C > 0$, for every $\ell \geq C \log n$, $\mathcal{E}_\delta(v, \ell) < \alpha^\ell$ for some constant $0 < \alpha < 1$.

It is sufficient to show that, for some constant integer $\theta > 0$ it holds that for every $v \in V$, every $L \geq C \log n$, every $P = (v, v_1, \ldots, v_{\theta L}) \in \mathsf{SAW}(v, \theta L)$, there exists a low-degree vertex v_j among $\{v_{L+1}, v_{\theta L}, \ldots, v_{\theta L}\}$.

Let $\theta = \max\left\{\left\lceil \log_{1/\alpha}\left(\frac{q-1}{2(1-\beta)}\right)\right\rceil, 2\right\}$. Assume for the contradiction that every vertex in $\{v_{L+1}, v_{L+2}, \ldots, v_{\theta L}\}$ has high-degree. Since $\theta L > L \geq C \log n$, we have $\prod_{i=1}^{\theta L} \delta(\deg(v_i)) \leq \alpha^{\theta L}$.

On the other hand, since $\delta(d) \geq \delta(0) = \frac{2(1-\beta)}{q-1}$, we have $\prod_{i=1}^{\theta L} \delta(\deg(v_i)) \geq \left(\frac{2(1-\beta)}{q-1}\right)^L$. This is a contradiction for our choice of θ. ◀

We now define the error of our estimation.

▶ **Definition 16.** Let $\Omega = (G, \Lambda, \sigma)$ with $G = (V, E)$ be an instance of Potts model, $v \in V \setminus \Lambda$ be a vertex, $x \in [q]$ be a color and $\ell \in \mathbb{Z}$ be an integer. Let $\hat{p}_{\Omega,v,x,\ell} := \mathtt{marg}(\Omega, v, x, \ell)$ be the value returned by our algorithm. We define

$$\mathcal{E}_{\Omega,v,\ell} := \max_{y \in [q]} \log\left(\frac{\hat{p}_{\Omega,v,y,\ell}}{\mathbf{Pr}_\Omega[c(v) = y]}\right)$$

with the convention $0/0 = 1$.

Let $\Omega = (G, \Lambda, \sigma)$ with $G = (V, E)$ be an instance of Potts model, $v \in V \setminus \Lambda$ be a vertex, $x \in [q]$ be a color and $\ell \in \mathbb{Z}$ be an integer. We can recursively identify each vertex in the computation tree of $\mathtt{marg}(\Omega, v, x, \ell)$ with a subtree of $T = T_{\mathrm{SAW}}(G, v)$:

- the root of the computation tree is identified with the root of T, i.e., the single vertex path (v);
- assuming the notations used in the description of Algorithm 2, if $\mathtt{marg}(\Omega, v, x, \ell)$ is identified with a subtree of T rooted at self-avoiding walk P, then for every $\rho \in \mathcal{F}(B(v))$ and $i \in [m]$, the routine $\mathtt{marg}(\Omega_i^\rho, v_i, \rho_i, \ell - P_i)$ is identified with the subtree of T rooted at the concatenation of P and P_i.

With this property, the following lemma relates our error of estimation to the error function defined before.

▶ **Lemma 17.** *Let $\Omega = (G, \Lambda, \sigma)$ with $G = (V, E)$ be an instance of Potts model, $v \in V \setminus \Lambda$ be a vertex, $x \in [q]$ be a color and $\ell \in \mathbb{Z}$ be an integer.*

Let S denote the set of vertices in T that can be identified to the leaves of the computation tree of `marg` (Ω, v, x, ℓ). *Let $T = T_{\mathrm{SAW}}(G, v)$, then we have*

$$\mathcal{E}_{\Omega,v,\ell} \leq \mathcal{E}_{T,S}.$$

The key to prove Lemma 17 is to establish the *one-step contraction* of $\mathcal{E}_{\Omega,v,\ell}$, as stated in the following lemma:

▶ **Lemma 18.** *Let $\Omega = (G, \Lambda, \sigma)$ with $G = (V, E)$ be an instance of Potts model, $v \in V \setminus \Lambda$ be a vertex and $\ell \in \mathbb{Z}$ be an integer. Let $B = B(v)$ the minimal permissive block in G containing v. Assume the edge boundary $\delta B = \{u_i v_i \mid i \in [m]\}$ where $v_i \notin B$.*

Then it holds that

$$\mathcal{E}_{\Omega,v,\ell} \leq \sum_{i=1}^{m} \frac{1-\beta}{q-1-(1-\beta)\deg_G(v_i)} \cdot \max_{\rho \in \mathcal{F}(B)} \mathcal{E}_{\Omega_i^\rho, v_i, \ell_i}$$

where Ω_i^ρ is defined in Section 3 and $\ell_i = \ell - |P_i|$ for the self-avoiding walk P_i chosen in Algorithm 2.

Proof. Let $\pi \in \mathcal{F}(B)$ be a coloring of the block B. We use $\hat{p}_{\Omega,B,\pi,\ell} =$ `marg-block` (Ω, B, π, ℓ) to denote the value return by our estimation algorithm for block marginals. Let x denote the color that achieves the maximum in the definition of $\mathcal{E}_{\Omega,v,\ell}$, we have

$$\begin{aligned}\mathcal{E}_{\Omega,v,\ell} &= \log\left(\frac{\hat{p}_{\Omega,v,x,\ell}}{\mathbf{Pr}_\Omega[c(v)=x]}\right) \leq \log\left(\frac{\sum_{\substack{\pi\in\mathcal{F}(B)\\ s.t.\ \pi(v)=x}} \hat{p}_{\Omega,B,\pi,\ell}}{\sum_{\substack{\pi\in\mathcal{F}(B)\\ s.t.\ \pi(v)=x}} \mathbf{Pr}_\Omega[c(B)=\pi]}\right)\\ &\leq \max_{\substack{\pi\in\mathcal{F}(B)\\ s.t.\ \pi(v)=x}} \log\left(\frac{\hat{p}_{\Omega,B,\pi,\ell}}{\mathbf{Pr}_\Omega[c(B)=\pi]}\right).\end{aligned}$$

By our algorithm, all the marginals in the recursion satisfies the upper bound in Lemma 11, it then follows from Lemma 10 that for every $\pi \in \mathcal{F}(B)$, it holds that

$$\log\left(\frac{\hat{p}_{\Omega,B,\pi,\ell}}{\mathbf{Pr}_\Omega[c(B)=\pi]}\right) \leq \sum_{i=1}^{m} \frac{2(1-\beta)}{q-(1-\beta)\deg_G(v_i)-1} \cdot \max_{\rho\in\mathcal{F}(B)} \mathcal{E}_{\Omega_i^\rho, v_i, \ell_i}.$$ ◀

We can use Lemma 18 to prove Lemma 17.

Proof of Lemma 17. We apply induction on $T := T_{\mathrm{SAW}}(G, v)$. The base case is that $v \in \Lambda$ or $v \in S$. If $v \in S$ and $\beta > 0$, then by Lemma 11 and Lemma 12, it holds that

$$\mathcal{E}_{\Omega,v,\ell} = \max_{y\in[q]} \log\left(\frac{\hat{p}_{\Omega,v,y,\ell}}{\mathbf{Pr}_\Omega[c(v)=y]}\right) \leq q + n\log\frac{1}{\beta}.$$

If $v \in S$ and $\beta = 0$, by Proposition 8 and Lemma 11, we have

$$\mathcal{E}_{\Omega,v,\ell} = \max_{y\in[q]} \log\left(\frac{\hat{p}_{\Omega,v,y,\ell}}{\mathbf{Pr}_\Omega[c(v)=y]}\right) \leq n\log q.$$

If $v \in \Lambda$, then $\mathcal{E}_{\Omega,v,\ell} = 0$.

Now assume $v \notin S \cup \Lambda$ and denote $B = B(v)$ the minimal permissive block containing v. Assume the edge boundary $\delta B = \{u_i v_i \mid u_i \in B\}$.

It then follows from Lemma 18 that

$$\mathcal{E}_{\Omega,v,\ell} \le \sum_{i=1}^{m} \frac{1-\beta}{q-1-(1-\beta)\deg_G(v_i)} \cdot \max_{\rho\in\mathcal{F}(B)} \mathcal{E}_{\Omega_i^\rho,v_i,\ell_i}.$$

Recall for every $i \in [m]$, we define in Algorithm 2 a self-avoiding walk P_i containing u_iv_i with every intermediate vertices in B. For every $u \in P_i$ such that $u \neq v, v_i$, it holds that $\delta(\deg_G(u)) = 1)$. With this property, if we use T_i to denote the subtree of T rooted at P_i, then

$$\sum_{i=1}^{m} \frac{1-\beta}{q-q-(1-\beta)\deg_G(v_i)} \cdot \mathcal{E}_{T_i,S} \le \mathcal{E}_{T,S}.$$

We can then complete the proof by using induction hypothesis to show

$$\mathcal{E}_{\Omega_i^\rho,v_i,\ell_i} \le \mathcal{E}_{T_i,S}$$

for every $i \in [m]$ and $\rho \in \mathcal{F}(B)$. ◀

We are now ready to prove the main lemma of this section.

Proof of Lemma 13

We can assume $v \notin \Lambda$, otherwise, the color on v is fixed by σ.

Let $T := T_{\mathrm{SAW}}(G, v)$. For every ℓ, let S_ℓ denote the set of vertices at which the procedure $\mathtt{marg}(\Omega, v, x, \ell)$ terminates. Then it follows Lemma 17,

$$\log\left(\frac{\hat{p}_{\Omega,v,x,\ell}}{\mathbf{Pr}_\Omega[c(v)=x]}\right) \le \mathcal{E}_{\Omega,v,\ell} \le \mathcal{E}_{T,S_\ell}.$$

Note that $\mathrm{dist}_T(v, S_\ell) \ge \ell$, since $\delta(\cdot)$ is a contraction function for $\mathcal{G}$, we have $\mathcal{E}_{T,S_\ell} \le n^C\gamma^\ell$ for some constants $C > 0$ and $0 < \gamma < 1$. This implies that for some constant $C_0 > 0$, if $\ell \ge C_0 \log n$, then

$$1 - O\left(\frac{1}{n^3}\right) \le \frac{\hat{p}_{\Omega,v,x,\ell}}{\mathbf{Pr}_\Omega[c(v)=x]} \le 1 + O\left(\frac{1}{n^3}\right).$$

To bound the running time of $\mathtt{marg}(\Omega, v, x, \ell)$, we can apply Lemma 15 to conclude that if $\ell = \Theta(\log n)$, then the algorithm must terminate at depth $L = \Theta(\log n)$ of T, i.e., for every $u \in S_\ell$, it holds that $\mathrm{dist}_T(v, u) \le L$.

We use T_ℓ to denote the subtree of T obtained by removing all descendants of S_ℓ and let $\mathcal{L}(T_\ell)$ to denote the set of self-avoiding walks corresponding to leaves of T_ℓ. Let $\tau_{\Omega,v,\ell}$ to denote the maximum running time of $\mathtt{marg}(\Omega, v, x, \ell)$ over all colorings $x \in [q]$, we apply induction on T_ℓ to show that for some $C_1 > 0$, it holds that

$$\tau_{\Omega,v,\ell} \le n^{C_1} \cdot \sum_{P\in\mathcal{L}(T_\ell)} q^{2|B(P)|}. \tag{4}$$

The base case is that $v \in S_\ell$ or $v \in \Lambda$ and our bound for running time trivially holds. Otherwise, denote $B = B_\Omega(v)$ the minimal permissive block containing v. Assume the edge boundary $\delta B = \{u_iv_i \mid u_i \in B\}$. We have for some constant $C_2 > 0$, it holds that

$$\tau_{\Omega,v,\ell} \le q^{|B_\Omega(v)|} n^{C_2} + q^{|B_\Omega(v)|} \sum_{i=1}^{m} \max_{\rho\in\mathcal{F}(B(v))} \tau_{\Omega_i^\rho,v_i,\ell_i} \tag{5}$$

Recall that P_i is a self-avoiding walk from v to v_i containing u_i with every intermediate vertex in B. Let T_i denote the subtree of T rooted at P_i. Then we can apply induction hypothesis to obtain

$$\tau_{\Omega_i^\rho, v_i, \ell_i} \le n^{C_1} \cdot \sum_{P \in \mathcal{L}(T_{\ell_i})} q^{2\left|B_{\Omega_i^\rho}(P)\right|} \tag{6}$$

for every $\rho \in \mathcal{F}(B)$ and $i \in [m]$. Furthermore, by our construction of Ω_i^ρ and the definition of permissive block, we have $|B_\Omega(v)| + \left|B_{\Omega_i^\rho}(v_i)\right| \le \left|B_\Omega(v) \cup B_{\Omega_i^\rho(v_i)}(v_i)\right|$ for every $\rho \in \mathcal{F}(B)$ and $i \in [m]$. Plugging (6) into (5) proves (4).

Since G is locally sparse, we know that the term $q^{2|B(P)|}$ is bounded by a polynomial in n for every $P \in \mathcal{L}(T_\ell)$. It remains to show that $|\mathcal{L}(T_\ell)|$ is bounded by a polynomial in n. To see this, consider the contribution of a walk P in $\mathcal{L}(T_\ell)$ of length k to the quantity $\mathcal{E}_\delta(v,k)$ defined in (1). The contribution of this walk is at least $\frac{1}{\mathrm{poly}(n)}$ since $k = O(\log n)$ and for every $u \in P$, the value $\delta(u)$ is bounded below by a constant. It then follows from the fact that $\delta(\cdot)$ is a contraction function for G, there are at most polynomial many leaves in T_ℓ for our choice of ℓ.

5 The FPTAS and the sampling algorithm

In this section, we prove Theorem 7, by using the correlation decay property established in Section 4.

Proof of Theorem 7. Let $\Omega = (G, \varnothing, \varnothing)$ be an instance of Potts model, where $G = (V,E) \in \mathcal{G}$. Without loss of generality, we give an algorithm to compute an approximation of the partition function $\hat{Z}(\Omega)$ satisfying

$$1 - O\left(\frac{1}{n^2}\right) \le \frac{\hat{Z}(\Omega)}{Z(\Omega)} \le 1 + O\left(\frac{1}{n^2}\right).$$

Since our family of instances of Potts model is "self-embeddable" in the sense of [22], the algorithm can be boosted into an FPTAS.

Assume $V = \{v_1, \dots, v_n\}$. First find a configuration $\sigma \in [q]^V$ such that $w_G(\sigma) > 0$. This task is trivial when $\beta > 0$. When $\beta = 0$, since G is q-colorable, we can also do it in polynomial time:

- If the graph is not empty, then choose a vertex v and find a feasible coloring of $B(v)$. Then remove $B(v)$ from the graph and repeat the process.

If G is q-colorable, then $G[V \setminus B(v)]$ is colorable as the boundary of $B(v)$ consists of low-degree vertices, thus the above process will end with a proper coloring of G, which is the union of colorings found at each step. The process terminates in polynomial time since $\mathcal{G}$ is locally sparse and thus the size of every $B(v)$ is $O(\log n)$.

With σ in hand, we have

$$\begin{aligned} Z(\Omega) = w_G(\sigma)/\mathbf{Pr}_\Omega\left[c(V) = \sigma\right] &= w_G(\sigma)\left(\mathbf{Pr}_\Omega\left[\bigwedge_{i=1}^n c(v_i) = \sigma(v_i)\right]\right)^{-1} \\ &= w_G(\sigma)\left(\prod_{i=1}^n \mathbf{Pr}_\Omega\left[c(v_i) = \sigma(v_i) \;\middle|\; \bigwedge_{j=1}^{i-1} c(v_j) = \sigma(v_j)\right]\right)^{-1} \end{aligned}$$

For every $i \in [n]$, let $\Omega_i = (G, \Lambda_i, \sigma_i)$ where $\Lambda_i = \{v_1, \dots, v_{i-1}\}$ and $\sigma_i(v_j) = \sigma(v_j)$ for every $j = 1, \dots, i-1$. We have

$$Z(\Omega) = w_G(\sigma) \left(\prod_{i=1}^{n} \mathbf{Pr}_{\Omega_i} [c(v_i) = \sigma(v_i)] \right)^{-1}.$$

Note that the graph class $\mathcal{G}$ is closed under the operation of fixing some vertex to a specific color, we can apply Lemma 13 for every Ω_i and obtain $\hat{p}_i$ such that

$$1 - O\left(\frac{1}{n^3}\right) \le \frac{\hat{p}_i}{\mathbf{Pr}_{\Omega_i} [c(v_i) = \sigma(v_i)]} \le 1 + O\left(\frac{1}{n^3}\right).$$

Let $\hat{Z}(\Omega) = w_G(\sigma) \left(\prod_{i=1}^{n} \hat{p}_i\right)^{-1}$, then Theorem 13 implies that

$$1 - O\left(\frac{1}{n^2}\right) \le \frac{\hat{Z}(\Omega)}{Z(\Omega)} \le 1 + O\left(\frac{1}{n^2}\right).$$

This approximate counting algorithm implies a sampling algorithm via Jerrum-Valiant-Vazirani reduction[17]. ◀

6 Random Graphs

In this section, we prove Theorem 2. We first prove the following properties of $\mathcal{G}(n, d/n)$.

▶ **Theorem 19.** *Let d be a sufficiently large constant, $q > 3(1-\beta)d + 4$ and $G = (V, E) \sim \mathcal{G}(n, d/n)$. Then with probability $1 - o(1)$, the following holds*

- *there exist two universal positive constants $C > 0, \gamma < 1$ such that $\mathcal{E}_\delta(v, \ell) < n^C \gamma^\ell$ for all $v \in V$ and for all $\ell = o(\sqrt{n})$, where $\mathcal{E}_\delta(v, \ell)$ is defined in (1);*
- *if $\beta = 0$, then G is q-colorable;*
- *there exists a universal constant $C > 0$ such that for every path P in G of length ℓ, $|B(P)| \le C(\ell + \log n)$.*

Note that the first property in above theorem impose an upper bound on ℓ. This is not harmful as our algorithms for FPTAS and sampling only require the property holds for $\ell = O(\log n)$. Thus Theorem 19 and Theorem 7 together imply Theorem 2.

It is well-known that when $\beta = 0$, G is q-colorable with high probability (see e.g., [12]), we verify the first property in Lemma 20 and the third property in Lemma 22.

6.1 Contraction function for random graphs

▶ **Lemma 20.** *Let $d > 1$, $0 \le \beta < 1$ and $q > 3(1-\beta)d + 4$ be constants. Let $G = (V, E) \sim \mathcal{G}(n, d/n)$. There exist two positive constants $C > 0$ and $\gamma < 1$ such that with probability $1 - O\left(\frac{1}{n}\right)$, for every $v \in V$ and every $\ell = o(\sqrt{n})$, it holds that*

$$\mathcal{E}_\delta(v, \ell) \le n^C \gamma^\ell$$

We first prove a technical lemma.

▶ **Lemma 21.** *Let $0 \le \beta < 1$ be a constant. Let $f_q(d) : \mathbb{R}^{\ge 0} \to \mathbb{R}^{\ge 0}$ be a piece wise function defined as*

$$f_q(d) := \begin{cases} \frac{2(1-\beta)}{q-1-(1-\beta)d} & \textit{if } d \le \frac{q-1}{1-\beta} - 2 \\ 1 & \textit{otherwise.} \end{cases}$$

Let X be a random variable distributed according to binomial distribution $\mathrm{Bin}(n, \frac{\Delta}{n})$ *where* $\Delta > 1$ *is a constant. Then for* $q \geq 3(1-\beta)\Delta + 2$ *and all sufficiently large* n*, it holds that* $\mathbf{E}\left[f_q(X)\right] < \frac{1}{\Delta}$.

Proof. Let $\lambda = 1 - \beta$. Since $f(d)$ is decreasing in q, we can assume $q = 3\lambda\Delta + 2$. Note that

$$\mathop{\mathbf{E}}_{d\sim\mathrm{Bin}\left(n,\frac{\Delta}{n}\right)}[f(d)] \leq \frac{1}{\Delta} \iff \mathop{\mathbf{E}}_{d\sim\mathrm{Bin}\left(n,\frac{\Delta}{n}\right)}[1 - f(d)] \geq \frac{\Delta - 1}{\Delta}.$$

Let $g(x) := 1 - f(x)$, then

$$\mathop{\mathbf{E}}_{d\sim\mathrm{Bin}\left(n,\frac{\Delta}{n}\right)}[1 - f(d)] = \sum_{k=0}^{\lfloor \frac{q-1}{\lambda} - 2 \rfloor} g(k) \cdot p(k)$$

where $p(k) = \binom{n}{k}\left(\frac{\Delta}{n}\right)^k\left(1 - \frac{\Delta}{n}\right)^{n-k}$.

Define

$$\begin{aligned}\tilde{g}(x) := 1 &- \frac{2\lambda}{q - 1 - \lambda\Delta} - \frac{2\lambda^2(x - \Delta)}{(q - 1 - \lambda\Delta)^2} - \frac{2\lambda^3(x - \Delta)^2}{(q - 1 - \lambda\Delta)^3} - \frac{2\lambda^4(x - \Delta)^3}{(q - 1 - \lambda\Delta)^4} \\ &- \frac{2\lambda^5(x - \Delta)^4}{(q - 1 - \lambda\Delta)^5} - \frac{2\lambda^6(x - \Delta)^5}{(q - 1 - \lambda\Delta)^6} - \frac{2\lambda^6(x - \Delta)^6}{(q - 1 - \lambda\Delta)^6}.\end{aligned}$$

Then

$$g(x) - \tilde{g}(x) = \frac{2\lambda^6(q - 1 - \lambda - x\lambda)(x - \Delta)^6}{(q - 1 - x\lambda)(q - 1 - \lambda\Delta)^6},$$

which is positive for $x \leq \lfloor \frac{q-1}{\lambda} - 2 \rfloor$.

We now prove that

$$\sum_{k=0}^{\lfloor \frac{q-1}{\lambda} - 2 \rfloor} \tilde{g}(k) \cdot p(k) \geq \frac{\Delta - 1}{\Delta}.$$

The expectation of $\tilde{g}(k)$ can be computed directly:

$$\mathbf{E}\left[\tilde{g}(k)\right] = \frac{1}{n^5(q - 1 - \lambda\Delta)^6} \cdot \left(C_5 n^5 + C_4 n^4 \pm O(n^3)\right),$$

where

$$\begin{aligned}C_5 &= 1 - 2\lambda + (12\lambda - 20\lambda^2 - 2\lambda^3 - 2\lambda^4 - 2\lambda^5 - 4\lambda^6)\Delta \\ &\quad + (60\lambda^2 - 80\lambda^3 - 12\lambda^4 - 14\lambda^5 - 74\lambda^6)\Delta^2 + (160\lambda^3 - 160\lambda^4 - 24\lambda^5 - 50\lambda^6)\Delta^3 \\ &\quad + (240\lambda^4 - 160\lambda^5 - 16\lambda^6)\Delta^4 + (192\lambda^5 - 64\lambda^6)\Delta^5 + 64\lambda^6\Delta^6; \\ C_4 &= 2\lambda^3(1 + 3\lambda + 7\lambda^2 + 46\lambda^3)\Delta^2 + 2\lambda^3(6\lambda + 18\lambda^2 + 234\lambda^3)\Delta^3 \\ &\quad + 2\lambda^3(12\lambda^2 + 69\lambda^3)\Delta^4 + 16\lambda^6\Delta^5.\end{aligned}$$

Since $C_4 > 0$, thus for sufficiently large n, it holds that

$$\mathbf{E}\left[\tilde{g}(x)\right] \geq \frac{C_5}{(q - 1 - \lambda\Delta)^6}.$$

We also have that

$$\mathbf{E}\left[\tilde{g}(x)\right] = \sum_{k=0}^{\lfloor \frac{q-1}{\lambda}-2 \rfloor} \tilde{g}(k)\cdot p(k) + \sum_{k=\lfloor \frac{q-1}{\lambda}-1 \rfloor}^{n} \tilde{g}(k)\cdot p(k)$$

It can be verified that $\tilde{g}(x)$ is monotonically decreasing in x when $x \geq \frac{q-1}{\lambda} - 2$ and $\tilde{g}\left(\frac{q-1}{\lambda}-2\right) = -\left(\frac{1+2\lambda(\Delta-1)}{1+2\lambda\Delta}\right)^6 < 0$.

Thus we have

$$\sum_{k=0}^{\lfloor \frac{q-1}{\lambda}-2 \rfloor} \tilde{g}(k)\cdot p(k) \geq \mathbf{E}\left[\tilde{g}(x)\right] \geq \frac{C_5}{(q-1-\lambda\Delta)^6} = \frac{\Delta-1}{\Delta} + h(\Delta)$$

where

$$\begin{aligned} h(\Delta) = \big(&1 + 10\lambda\Delta + (40\lambda^2 - 2\lambda^3 - 2\lambda^4 - 2\lambda^5 - 4\lambda^6)\Delta^2 \\ &+(80\lambda^3 - 12\lambda^4 - 14\lambda^5 - 74\lambda^6)\Delta^3 + (80\lambda^4 - 24\lambda^5 - 50\lambda^6)\Delta^4\big) \\ &+(32\lambda^5 - 16\lambda^6)\Delta^5\big)\cdot\big(\Delta(1+2\lambda\Delta)^6\big)^{-1}. \end{aligned}$$

It can be verified that $h(\Delta)$ is positive for every $0 < \lambda < 1$ and $\Delta \geq 1$. ◀

Proof of Lemma 20. Let $v \in V$ be arbitrary fixed and $T_v = T_{\text{SAW}}(G, v)$ and $\ell > 0$ be an integer. By linearity of expectation, we have

$$\mathbf{E}\left[\mathcal{E}_\delta(v,\ell)\right] \leq n^\ell \left(\frac{d}{n}\right)^\ell \mathbf{E}\left[\prod_{i=1}^{\ell} \delta(\deg_G(v_i)) \,\middle|\, P = (v, v_1, \ldots, v_\ell) \text{ is a path}\right].$$

Fix a tuple $P = (v, v_1, \ldots, v_\ell)$. To calculate the expectation, we construct an independent sequence whose product dominates $\prod_{i=1}^{\ell} \delta(\deg_G(v_i))$ as follows.

Conditioning on $P = (v, v_1, \ldots, v_\ell)$ being a path in G. Let $X_1, X_2, \ldots, X_\ell$ be random variables such that each X_i represents the number of edges between v_i and vertices in $V \setminus \{v_1, \ldots, v_\ell\}$; and let Y be a random variable representing the number of edges between vertices in $\{v_1, \ldots, v_\ell\}$ except for the edges in the path $P = (v, v_1, \ldots, v_\ell)$. Then $X_1, \ldots, X_\ell, Y$ are mutually independent binomial random variables with each X_i distributed according to $\text{Bin}(n-\ell, \frac{d}{n})$ and Y distributed according to $\text{Bin}(\binom{\ell}{2} - \ell + 1, \frac{d}{n})$, and for each v_i in the path we have $\deg_G(v_i) = X_i + 2 + Y_i$ with some $Y_1 + Y_2 + \cdots + Y_\ell = 2Y$.

Note that $\delta(\deg_G(v_i)) = f_q(\deg_G(v_i))$ where the function $f_q(x)$ is defined in Lemma 21. Note that the ratio $f_q(x)/f_q(x-1)$ is always upper bounded by 2, and we have $f_q(x+1) \leq f_{q-1}(x)$. Thus, conditioning on that $P = (v, v_1, \ldots, v_\ell)$ is a path, the product $\prod_{i=1}^{\ell} \delta_{q,\beta}(\deg_G(v_i))$ can be bounded as follows:

$$\prod_{i=1}^{\ell} \delta(\deg_G(v_i)) = \prod_{i=1}^{\ell} f_q(X_i + Y_i + 2) \leq 2^{2Y} \prod_{i=1}^{\ell} f_{q-2}(X_i).$$

Let $d' = \frac{q-4}{3(1-\beta)}$, then we have $d' > d$. Let X be a binomial random variable distributed according to $\text{Bin}(n, \frac{d'}{n})$, thus X probabilistically dominates every X_i whose distribution is $\text{Bin}(n-\ell, \frac{d}{n})$. Since $X_1, X_2, \ldots, X_\ell, Y$ are mutually independent conditioning on $P = (v, v_1, \ldots, v_\ell)$ being a path in G, for any $P = (v, v_1, \ldots, v_\ell)$ we have

$$\mathbf{E}\left[\prod_{i=1}^{\ell} \delta(\deg_G(v_i)) \,\middle|\, P \text{ is a path}\right] \leq \mathbf{E}\left[4^Y \prod_{i=1}^{\ell} f_{q-2}(X_i)\right] \leq \mathbf{E}\left[4^Y\right] \mathbf{E}\left[f_{q-2}(X)\right]^\ell.$$

Recall that $Y \sim \mathrm{Bin}\left(\binom{\ell}{2} - \ell + 1, \frac{d}{n}\right)$, the expectation $\mathbf{E}\left[4^Y\right]$ can be bounded as

$$\mathbf{E}\left[4^Y\right] \le \sum_{k=0}^{\ell^2} 4^k \binom{\ell^2}{k} \left(\frac{d}{n}\right)^k \left(1 - \frac{d}{n}\right)^{\ell^2 - k} = \left(1 + \frac{3d}{n}\right)^{\ell^2} \le \exp\left(\frac{3d\ell^2}{n}\right).$$

Since $q - 2 \ge 3(1-\beta)d' + 2$, it follows from Lemma 21 that $\mathbf{E}\left[f_{q-2}(X)\right] \le \frac{1}{d'} = \frac{3(1-\beta)}{q-4}$. Therefore,

$$\begin{aligned}\mathbf{E}\left[\prod_{i=1}^{\ell} \delta(\deg_G(v_i)) \,\middle|\, P \text{ is a path}\right] &\le \exp\left(\frac{3d\ell^2}{n}\right)\left(\frac{3(1-\beta)}{q-4}\right)^{\ell} \\ &\le \frac{1}{d^{\ell}} \cdot \exp\left(-\ell \log\left(\frac{q-4}{3d(1-\beta)}\right) + \frac{3d\ell^2}{n}\right).\end{aligned}$$

Since $\ell = o(\sqrt{n})$,

$$\mathbf{E}\left[\mathcal{E}_{\delta}(v,\ell)\right] \le \exp\left(-\ell \log\left(\frac{q-4}{3d(1-\beta)}\right) + o(1)\right).$$

Then the lemma follows from the Markov inequality and the union bound. ◀

6.2 Locally sparsity for random graph

▶ **Lemma 22.** *Let $\varepsilon > 0$ be some fixed constant. Let d be a sufficiently large number, $q \ge (2+\varepsilon)d$ and $0 \le \beta < 1$ be constants. Let $G = (V,E) \sim \mathcal{G}(n, d/n)$. There exists a constant $C > 0$ such that with probability $1 - O\left(\frac{1}{n}\right)$, for every path P in G of length ℓ, $|B(P)| \le C(\ell + \log n)$.*

Given $P = (v_1, \dots, v_L)$, we are going to upper bound the probability

$$\mathbf{Pr}\left[|B(P)| \ge t \mid P \text{ is a path}\right] \tag{7}$$

for every $t > 0$.

A vertex v is a high-degree vertex if $\deg_G(v) \ge \frac{q-1}{1-\beta} - 2$. Thus the probability (7) is maximized when $\beta = 0$. Note that conditioning on P is a path gives each vertex at most two degrees, we can redefine the notion of "high-degree" as $\deg_G(v) \ge q - 5$ and drop the condition that P is a path. Thus it is sufficient to upper bound

$$\mathbf{Pr}\left[|B(P)| \ge t\right]$$

with our new definition of high-degree vertices.

Let $G = (V,E)$ be a graph. We now describe a BFS procedure to generate $B^*(P) := B(P) \cup \partial B(P)$. Since $B^*(P)$ is always a superset of $B(P)$, it is sufficient to bound $\mathbf{Pr}\left[|B^*(P)| \ge t\right]$. For a vertex $v \in V$, we use $N_G(v)$ to denote the set of neighbors of v in G.

Initially, we have a counter $i = 0$, a graph $G_0 = G$, a set of active vertices $\mathcal{A}_0 = \{v_1, v_2, \dots, v_L\}$ and a set of used vertices $\mathcal{U}_0 = \varnothing$.

(P1)

1. Increase the counter i by one.
2. (If $i \le L$) Define $G_i(V_i, E_i) = G_{i-1}[V_{i-1} \setminus \{v_i\}]$. Let $\mathcal{U}_i = \mathcal{U}_{i-1} \cup \{v_i\}$. Let $\mathcal{A}_i = (\mathcal{A}_{i-1} \cup N_{G_{i-1}}(v_i)) \setminus \mathcal{U}_i$. Goto 1.

3. (If $i > L$) Terminate if $\mathcal{A}_{i-1} = \varnothing$. Otherwise, let $u \in \mathcal{A}_{i-1}$ and let $\mathcal{U}_i = \mathcal{U}_{i-1} \cup \{u\}$.
   a. (If $|N_G(u)| \geq q-5$) Define $G_i(V_i, E_i) = G_{i-1}[V_{i-1} \setminus \{u\}]$. Let $\mathcal{A}_i = (\mathcal{A}_{i-1} \cup N_{G_{i-1}}(v_i)) \setminus \mathcal{U}_i$. Goto 1.
   b. (If $|N_G(u)| < q-5$) Define $G_i = G_{i-1}$. Let $\mathcal{A}_i = \mathcal{A}_{i-1} \setminus \mathcal{U}_i$. Goto 1

The following proposition is immediate:

▶ **Proposition 23.** *Assume the algorithm terminates at step t, then $B^*(P) = \mathcal{U}_{t-1}$ and $|B^*(P)| = t - 1$.*

Let $R = \{r_1, r_2, \ldots, r_L\}$ be a set and each r_i is the root of tree T_i. We now describe a BFS procedure to explore these L trees. For a vertex v, we use $C(v)$ to denote its children.

Initially, we have a counter $i = 0$ and a set of active vertices $\mathcal{B}_0 = R$.

(P2)

1. Increase the counter i by one.
2. (If $i \leq L$) Let $\mathcal{B}_i = (\mathcal{B}_{i-1} \cup C(r_i)) \setminus \{r_i\}$. Goto 1.
3. (If $i > L$) Terminate if $\mathcal{B}_{i-1} = \varnothing$. Otherwise, let $w \in \mathcal{B}_{i-1}$
   a. (If $|C(w)| \geq \frac{q-5}{2}$) Let $\mathcal{B}_i = (\mathcal{B}_{i-1} \cup C(u)) \setminus \{w\}$. Goto 1.
   b. (If $|C(w)| < \frac{q-5}{2}$) Let $\mathcal{B}_i = \mathcal{B}_{i-1} \setminus \{w\}$. Goto 1.

Now assume $G \sim \mathcal{G}(n, d/n)$ and for every $i \in [L]$, T_i is a branching process with distribution $\mathrm{Bin}(n, d/n)$, i.e., each $C(u) \sim \mathrm{Bin}([n], d/n)$. We can implement the (P1) when at each step i, the vertex u chosen from the active set sample its neighbors $N_{G_{i-1}}(u)$ according to $\mathrm{Bin}(V_i, d/n)$. This random process can be coupled with $\mathcal{G}(n, d/n)$ such that $B^*(P)$ found by it is always a superset of the one in $\mathcal{G}(n, d/n)$.

We now construct a coupling of (P1) and (P2) with the property that the later one always terminates no earlier than the former one.

At each step $i \geq 1$, let u and w be the vertex chosen from $\mathcal{A}_i$ and $\mathcal{B}_i$ respectively ($u = v_i$ and $w = r_i$ if $i \leq L$). Then $\left|N_{G_{i-1}}(u)\right| \sim \mathrm{Bin}(|V_i|, d/n)$. We couple it with some $x \sim \mathrm{Bin}(n, d/n)$ with the property that $x \geq \left|N_{G_{i-1}}(v_i)\right|$ and let $C(w)$ be a set with x elements.

▶ **Lemma 24.** *For every $i \geq 0$, the following two properties hold:*

(i1) *There exists a surjective mapping F_i from $\mathcal{B}_i$ to $\mathcal{A}_i$ in each step i.*

(i2) *For every $u \in \mathcal{A}_i$, we use $n_i(u)$ to denote the number of $w \in \mathcal{B}_i$ such that $F_i(w) = u$. Then for every $u \in \mathcal{A}_i$, $n_i(u) \geq |N_G(u)| - |N_{G_i}(u)|$.*

Proof. We apply induction on i to prove the lemma.

When $i = 0$, we let $F_0 : \mathcal{B}_0 \to \mathcal{A}_0$ be the function that $F_0(r_j) = v_j$ for every $j \in [L]$. Then both properties hold trivially.

Assume the lemma holds for smaller i. If $i \leq L$, since by our coupling, $|C(r_i)| \geq \left|N_{G_{i-1}}(v_i)\right|$, we can construct F_i by extending F_{i-1} with an arbitrary surjective mapping from $C(r_i)$ to $N_{G_{i-1}}(v_i)$. For every $u' \in \mathcal{A}_i$, if $u' \in N_{G_{i-1}}(v_i)$, then $n_i(u') \geq n_{i-1}(u') + 1$ and $\left|N_{G_{i-1}(u)}\right| - |N_{G_i}(u')| = 1$; otherwise $n_i(u') = n_{i-1}(u')$ and $\left|N_{G_{i-1}}(u)\right| = |N_{G_i}(u')|$. Induction hypothesis implies both $(i1)$ and $(i2)$ hold.

If $i > L$, we have to distinguish between cases:

- (If $|N_G(u)| \geq q-5$ and $N_{G_{i-1}}(u) \geq \frac{q-5}{2}$) We construct F_i by extending F_{i-1} with an arbitrary surjective mapping from $C(w)$ to $N_{G_{i-1}}(u)$, the same argument as $i \leq L$ case proves $(i1)$ and $(i2)$.

- (If $|N_G(u)| \geq q-5$ and $N_{G_{i-1}}(u) < \frac{q-5}{2}$) In this case, by induction hypothesis, we know that
$$n_{i-1}(u) \geq |N_G(u)| - \left|N_{G_{i-1}}(u)\right| \geq \frac{q-5}{2} > N_{G_{i-1}}(u).$$
Choose a surjective f from $F_{i-1}^{-1}(u)$ to $N_{G_{i-1}}(u)$ and construct F_i from F_{i-1} by replacing the mapping on $F_{i-1}^{-1}(u)$ by f. This is safe since $u \notin \mathcal{A}_i$. The same argument as before proves $(i1)$ and $(i2)$.
- (If $|N_G(u)| < q-5$) Construct $F_i = F_{i-1}$. Since everything does not change, the induction hypothesis implies $(i1)$ and $(i2)$. ◀

The first property above guarantees that (P2) terminates no earlier than (P1) and thus its stopping time is an upper bound for the size of $B^*(P)$ found by (P1).

(P2) can be modeled as follows:

1. Let $X \sim \text{Bin}(n, d/n)$ and $X_1, X_2 \ldots$ be an infinite sequence of independent random variables defined as follows
   - For $i = 1, 2, \ldots, L$, X_i is an independent copy of X;
   - For $i > L$, X_i has following distribution
$$X_i = \begin{cases} 0 & \text{if } X < (q-5)/2 \\ X & \text{otherwise.} \end{cases}$$
2. $Y_1, Y_2, \ldots$ is an infinite sequence of random variables that $Y_0 = L$ and $Y_i = Y_{i-1} + X_i - 1$ for every $i \geq 1$.
3. $Z = \min_t \{Y_t = 0\}$.

The above process is identical to (P2), thus we have

▶ **Proposition 25.** (P2) *terminates after step t if and only if $Z > t$.*

Note that $Z > t$ implies $Y_t \geq 0$, we turn to bound the latter.

▶ **Lemma 26.** *There exist two constants $C_1, C_2 > 0$ depending on d and ε such that*
$$\mathbf{Pr}\left[Y_t \geq 0\right] \leq \exp\left(-C_1 t + C_2 L\right).$$

Proof. By the definition, $Y_{t+L} = L - (t+L) + \sum_{i=1}^{L+t} X_i = -t + \sum_{i=1}^{L} X_i + \sum_{i=L+1}^{L+t} X_i$. We know the distribution of X_is and we now compute their moment generating function. For every $s > 0$, it holds that
$$\mathbf{Pr}\left[Y_{t+L} \geq 0\right] = \mathbf{Pr}\left[e^{sY_{t+L}} \geq 1\right] \leq \mathbf{E}\left[e^{sY_{t+L}}\right] = e^{-st}\left(\mathbf{E}\left[e^{sX}\right]\right)^L \left(\mathbf{E}\left[e^{sX_{L+1}}\right]\right)^t.$$
Recall that $X \sim \text{Bin}(n, d/n)$, we have $\mathbf{E}\left[e^{sX}\right] = \left(1 + \frac{d}{n}(e^s - 1)\right)^n \leq e^{d(e^s-1)}$. Let $p = (q-5)/2$, we have
$$\begin{aligned}\mathbf{E}\left[e^{sX_{L+1}}\right] &= \mathbf{Pr}\left[X < p\right] + \sum_{k=\lfloor p \rfloor}^{n} e^{sk} \cdot \mathbf{Pr}\left[X = k\right] \\ &\leq 1 + \sum_{k=\lfloor p \rfloor}^{n} e^{sk} \cdot \mathbf{Pr}\left[X \geq k\right] \\ &\leq \exp\left(\sum_{k=\lfloor p \rfloor}^{\infty} e^{sk} \cdot \mathbf{Pr}\left[X \geq k\right]\right)\end{aligned}$$

By Chernoff bound, for sufficiently large d, we have for some choices of $s > 0$ and $C_1 > 0$,

$$\sum_{k=\lfloor p \rfloor}^{\infty} e^{sk} \cdot \mathbf{Pr}\left[X \geq k\right] - s < -C_1'.$$

Let $C_2' = d(e^s - 1)$, we have

$$\mathbf{Pr}\left[Y_{t+L} \geq 0\right] \leq \exp\left(-C_1' t + C_2' L\right).$$

This implies for some constants $C_1, C_2 > 0$,

$$\mathbf{Pr}\left[Y_t \geq 0\right] \leq \exp\left(-C_1 t + C_2 L\right). \qquad ◀$$

Proof of Lemma 22. By Lemma 26 and the union bound, the probability that there exists a path P in G of length ℓ such that $|B(P)| \geq t$ is upper bounded by

$$n \cdot n^{\ell} \cdot \left(\frac{d}{n}\right)^{\ell} \cdot \mathbf{Pr}\left[Y_t \geq 0\right] \leq n \cdot d^{\ell} \cdot \exp\left(-C_1 t + C_2 \ell\right) = O\left(\frac{1}{n}\right)$$

for $t = C(\ell + \log n)$ and sufficiently large constant C. ◀

Acknowledgement. This work was done in part while both authors were visiting the Simons Institute for the Theory of Computing.

References

1 Russ Bubley and Martin Dyer. Path coupling: A technique for proving rapid mixing in Markov chains. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS'97)*, pages 223–231. IEEE, 1997.

2 Martin Dyer, Abraham Flaxman, Alan Frieze, and Eric Vigoda. Randomly coloring sparse random graphs with fewer colors than the maximum degree. *Random Structures & Algorithms*, 29(4):450–465, 2006.

3 Martin Dyer and Alan Frieze. Randomly coloring graphs with lower bounds on girth and maximum degree. *Random Structures & Algorithms*, 23(2):167–179, 2003.

4 Martin Dyer, Alan Frieze, Thomas Hayes, and Eric Vigoda. Randomly coloring constant degree graphs. *Random Structures & Algorithms*, 43(2):181–200, 2013.

5 Charilaos Efthymiou. A simple algorithm for random colouring $G(n, d/n)$ using $(2+\varepsilon)$d colours. In *Proceedings of the 23th Annual ACM-SIAM symposium on Discrete Algorithms (SODA'12)*, pages 272–280. SIAM, 2012.

6 Charilaos Efthymiou. Mcmc sampling colourings and independent sets of $G(n, d/n)$ near uniqueness threshold. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 305–316. SIAM, 2014.

7 Charilaos Efthymiou. Switching colouring of $G(n, d/n)$ for sampling up to Gibbs uniqueness threshold. In *In Proceedings of the 22nd European Symposium on Algorithms (ESA'14)*, pages 371–381. Springer, 2014.

8 Charilaos Efthymiou and Paul Spirakis. Random sampling of colourings of sparse random graphs with a constant number of colours. *Theoretical Computer Science*, 407(1):134–154, 2008.

9 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability for antiferromagnetic spin systems in the tree nonuniqueness region. *Journal of the ACM (JACM)*, 62(6):50, 2015.

10 David Gamarnik and Dmitriy Katz. Correlation decay and deterministic FPTAS for counting colorings of a graph. *Journal of Discrete Algorithms*, 12:29–47, 2012.

11 David Gamarnik, Dmitriy Katz, and Sidhant Misra. Strong spatial mixing of list coloring of graphs. *Random Structures & Algorithms*, 46(4):599–613, 2015.

12 Geoffrey Grimmett and Colin McDiarmid. On colouring random graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 77, pages 313–324. Cambridge Univ Press, 1975.

13 Thomas Hayes. Randomly coloring graphs of girth at least five. In *Proceedings of the 35th Annual ACM Symposium on Symposium on Theory of Computing (STOC'03)*, pages 269–278. ACM, 2003.

14 Thomas Hayes and Eric Vigoda. A non-markovian coupling for randomly sampling colorings. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pages 618–627. IEEE, 2003.

15 Thomas Hayes and Eric Vigoda. Coupling with the stationary distribution and improved sampling for colorings and independent sets. *The Annals of Applied Probability*, 16(3):1297–1318, 2006.

16 Mark Jerrum. A very simple algorithm for estimating the number of k-colorings of a low-degree graph. *Random Structures and Algorithms*, 7(2):157–166, 1995.

17 Mark Jerrum, Leslie Valiant, and Vijay Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

18 Johan Jonasson. Uniqueness of uniform random colorings of regular trees. *Statistics & Probability Letters*, 57(3):243–248, 2002.

19 Pinyan Lu and Yitong Yin. Improved FPTAS for multi-spin systems. In *Proceedings of APPROX-RANDOM*, pages 639–654. Springer, 2013.

20 Michael Molloy. The Glauber dynamics on colorings of a graph with high girth and maximum degree. *SIAM Journal on Computing*, 33(3):721–737, 2004.

21 Elchanan Mossel and Allan Sly. Gibbs rapidly samples colorings of $G(n, d/n)$. *Probability theory and related fields*, 148(1-2):37–69, 2010.

22 Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82(1):93–133, 1989.

23 Alistair Sinclair, Piyush Srivastava, Daniel Štefankovič, and Yitong Yin. Spatial mixing and the connective constant: Optimal bounds. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*, pages 1549–1563. SIAM, 2015.

24 Alistair Sinclair, Piyush Srivastava, and Yitong Yin. Spatial mixing and approximation algorithms for graphs with bounded connective constant. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS'13)*, pages 300–309. IEEE, 2013.

25 Eric Vigoda. Improved bounds for sampling colorings. *Journal of Mathematical Physics*, 41(3):1555–1569, 2000.

26 Yitong Yin. Spatial mixing of coloring random graphs. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP'14, Track A)*, pages 1075–1086. Springer, 2014.