Prasad Raghavendra Sofya Raskhodnikova
Klaus Jansen José D.P. Rolim (Eds.)

Approximation, Randomization, and Combinatorial Optimization

Algorithms and Techniques

LNCS 8096

16th International Workshop, APPROX 2013
and 17th International Workshop, RANDOM 2013
Berkeley, CA, USA, August 2013, Proceedings

Springer

Lecture Notes in Computer Science 8096

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Prasad Raghavendra Sofya Raskhodnikova
Klaus Jansen José D.P. Rolim (Eds.)

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

Springer

Volume Editors

Prasad Raghavendra
University of California
Berkeley, CA, USA
E-mail: prasad@cs.berkeley.edu

Sofya Raskhodnikova
Pennsylvania State University
University Park, PA, USA
E-mail: sofya@cse.psu.edu

Klaus Jansen
University of Kiel
Kiel, Germany
E-mail: kj@informatik.uni-kiel.de

José D.P. Rolim
University of Geneva
Carouge, Switzerland
E-mail: jose.rolim@unige.ch

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-40327-9 e-ISBN 978-3-642-40328-6
DOI 10.1007/978-3-642-40328-6
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013944978

CR Subject Classification (1998): F.2.2, G.2.2, G.2.1, F.1.2, G.1.0, G.1.2, G.1.6, G.3, E.1

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the papers presented at the 16th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2013) and the 17th International Workshop on Randomization and Computation (RANDOM 2013), which took place concurrently in UC Berkeley, during August 21–23, 2013. APPROX focuses on algorithmic and complexity issues surrounding the development of efficient approximate solutions to computationally difficult problems, and was the 16th 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), and Boston (2012). RANDOM is concerned with applications of randomness to computational and combinatorial problems, and was the 17th 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), and Boston (2012).

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, mathematical programming methods, combinatorial problems in graphs and networks, game theory, markets and economic applications, geometric problems, packing, covering, scheduling, approximate learning, design and analysis of online algorithms, 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 23 papers, selected by the APPROX Program Committee out of 46 submissions, and 25 papers, selected by the RANDOM Program Committee out of 52 submissions. In addition to presentations on these papers, the program included invited talks by Persi Diaconis (Stanford University, USA), Luca Trevisan (Stanford University, USA), and Santosh Vempala (Georgia Tech, USA).

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 support from the Computer Science Division,

University of California, Berkeley, the Department of Computer Science and Engineering at the Pennsylvania State University, the Institute of Computer Science of the Christian-Albrechts-Universität zu Kiel and the Department of Computer Science of the University of Geneva.

August 2013

Prasad Raghavendra
Sofya Raskhodnikova
Klaus Jansen
José D.P. Rolim

Organization

Program Committees

APPROX 2013

Nikhil Bansal	Eindhoven University, The Netherlands
Chandra Chekuri	University of Illinois, Urbana-Champaign, USA
Eden Chlamtáč	Ben-Gurion University, Israel
Nikhil Devanur	Microsoft Research Redmond, USA
Uriel Fiege	Weizmann Institute, Israel
Claire Matheiu	Brown University, USA
Ankur Moitra	Institute of Advanced Study, Princeton, USA
Seffi Naor	Technion, Israel
Yuval Rabani	Hebrew University, Israel
Prasad Raghavendra	University of California, Berkeley, USA (Chair)
Roy Schwartz	Microsoft Research Redmond, USA
Mohit Singh	Microsoft Research Redmond, USA
Ola Svennson	École Polytechnique Fédéral de Lausanne, Switzerland
Mohammed Taghi Hajighayi	University of Maryland, College Park, USA
Madhur Tulsiani	Toyota Technical Institute Chicago, USA
Rico Zenklusen	John Hopkins University, USA

RANDOM 2013

Amit Chakrabarti	Dartmouth College, USA
Nikolaos Fountalakis	University of Birmingham, UK
Ariel Gabizon	Technion, Israel
Parikshit Gopalan	Microsoft Research, Silicon Valley, USA
Dan Gutfreund	IBM Research, Haifa, Israel
Prahladh Harsha	Tata Institute of Fundamental Research, India
Tomas Hayes	University of New Mexico, USA
Michael Krivelevich	Tel Aviv University, Israel
Shachar Lovett	University of California at San Diego, USA
Russell Martin	University of Liverpool, UK
Dieter van Melkebeek	University of Wisconsin-Madison, USA
Sofya Raskhodnikova	Pennsylvania State University, USA (Chair)
Shubhangi Saraf	Rutgers University, USA
Christian Sohler	TU Dortmund University, Germany
David P. Woodruff	IBM Research, Almaden, USA
Amir Yehudayoff	Technion, Israel

External Reviewers

Melika Abolhasani
Matthew Anderson
Antonios Antoniadis
Per Austrin
Yossi Azar
Mohammadhossein Bateni
Tugkan Batu
Arnab Bhattacharyya
Abhishek Bhowmick
Eric Blais
Michel Bode
Beate Bollig
Magnus Bordewich
Joshua Brody
Shahar Chen
Amin Coja-Oghlan
Graham Cormode
Nicholas Crawford
Varsha Dani
Anindya De
Martin Dietzfelbinger
Michael Dinitz
Anne Driemel
Andrew Drucker
Zeev Dvir
Ebrahim Ehsanfar
David Eppstein
Moran Feldman
Michael Forbes
Tom Friedetzky
Alan Frieze
David Galvin
David Gamarnik
Sumit Ganguly
Ran Gelles
Shayan Oveis Gharan
Elena Grigorescu
Sudipto Guha
Ankit Gupta
Patrick Hayden
Frank Hellweg
Piotr Indyk
Rahul Jain
Tali Kaufman
Shiva Kasiviswanathan
Neeraj Kayal
Gillat Kol
Guy Kortsarz
Dariusz Kowalski
Lap Chi Lau
Massimo Lauria
James Lee
Jon Lee
Virginie Lerays
Ke Li
Kostya Makarychev
David Malec
Yishay Mansour
Kevin Matulef
Andrew McGregor
Raghu Meka
Morteza Monemizadeh
Cris Moore
Benjamin Moseley
Wolfgang Mulzer
Alexander Munteanu
Viswanath Nagarajan
Ilan Newman Krzysztof Onak
Rasmus Pagh
Debmalya Panigrahi
Will Perkins
Preyas Popat
Eric Price
Ilya Razenshteyn
Ricardo Restrepo Lopez
Noga Ron-Zewi
Aaron Roth
Guy Rothblum
Atri Rudra
Sushant Sachdeva
Rahul Santhanam
Swagato Sanyal
Ramprasad Saptharishi
Melanie Schmidt
Chris Schwiegelshohn
Pranab Sen

Ronen Shaltiel
Makrand Sinha
Gregory Sorkin
Perla Sousi
Piyush Srivastava
Alexandre Stauffer
Chaitanya Swamy
Ning Tan
Amnon Ta-Shma
Kunal Talwar
Prasad Tetali
Mikkel Thorup
Greg Valiant
Paul Valiant
Danny Vilenchik
Emanuele Viola
Omri Weinstein
Benjamin Weitz
Udi Wieder
Ryan Williams
Sergey Yekhanin
Yuichi Yoshida
Qin Zhang

Table of Contents

APPROX

Spectral Sparsification in Dynamic Graph Streams 1
Kook Jin Ahn, Sudipto Guha, and Andrew McGregor

The Online Stochastic Generalized Assignment Problem 11
Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat

On the NP-Hardness of Approximating Ordering Constraint Satisfaction Problems 26
Per Austrin, Rajsekar Manokaran, and Cenny Wenner

Approximating Large Frequency Moments with Pick-and-Drop Sampling 42
Vladimir Braverman and Rafail Ostrovsky

Generalizing the Layering Method of Indyk and Woodruff: Recursive Sketches for Frequency-Based Vectors on Streams 58
Vladimir Braverman and Rafail Ostrovsky

Capacitated Network Design on Undirected Graphs 71
Deeparnab Chakrabarty, Ravishankar Krishnaswamy, Shi Li, and Srivatsan Narayanan

Scheduling Subset Tests: One-Time, Continuous, and How They Relate 81
Edith Cohen, Haim Kaplan, and Yishay Mansour

On the Total Perimeter of Homothetic Convex Bodies in a Convex Container 96
Adrian Dumitrescu and Csaba D. Tóth

Partial Interval Set Cover – Trade-Offs between Scalability and Optimality 110
Katherine Edwards, Simon Griffiths, and William Sean Kennedy

Online Square-into-Square Packing 126
Sándor P. Fekete and Hella-Franziska Hoffmann

Online Non-clairvoyant Scheduling to Simultaneously Minimize All Convex Functions 142
Kyle Fox, Sungjin Im, Janardhan Kulkarni, and Benjamin Moseley

Shrinking Maxima, Decreasing Costs: New Online Packing and Covering Problems 158
Pierre Fraigniaud, Magnús M. Halldórsson, Boaz Patt-Shamir, Dror Rawitz, and Adi Rosén

Multiple Traveling Salesmen in Asymmetric Metrics 173
Zachary Friggstad

Approximate Indexability and Bandit Problems with Concave Rewards and Delayed Feedback 189
Sudipto Guha and Kamesh Munagala

The Approximability of the Binary Paintshop Problem 205
Anupam Gupta, Satyen Kale, Viswanath Nagarajan, Rishi Saket, and Baruch Schieber

Approximation Algorithms for Movement Repairmen 218
MohammadTaghi Hajiaghayi, Rohit Khandekar, M. Reza Khani, and Guy Kortsarz

Improved Hardness of Approximating Chromatic Number 233
Sangxia Huang

A Pseudo-approximation for the Genus of Hamiltonian Graphs 244
Yury Makarychev, Amir Nayyeri, and Anastasios Sidiropoulos

A Local Computation Approximation Scheme to Maximum Matching ... 260
Yishay Mansour and Shai Vardi

Sketching Earth-Mover Distance on Graph Metrics 274
Andrew McGregor and Daniel Stubbs

Online Multidimensional Load Balancing 287
Adam Meyerson, Alan Roytman, and Brian Tagiku

A New Regularity Lemma and Faster Approximation Algorithms for Low Threshold Rank Graphs 303
Shayan Oveis Gharan and Luca Trevisan

Interdiction Problems on Planar Graphs 317
Feng Pan and Aaron Schild

RANDOM

Conditional Random Fields, Planted Constraint Satisfaction and Entropy Concentration 332
Emmanuel Abbe and Andrea Montanari

Finding Heavy Hitters from Lossy or Noisy Data 347
Lucia Batman, Russell Impagliazzo, Cody Murray, and Ramamohan Paturi

Private Learning and Sanitization: Pure vs. Approximate Differential Privacy 363
Amos Beimel, Kobbi Nissim, and Uri Stemmer

Phase Coexistence and Slow Mixing for the Hard-Core Model on $\mathbb{Z}^2$ 379
Antonio Blanca, David Galvin, Dana Randall, and Prasad Tetali

Fast Private Data Release Algorithms for Sparse Queries 395
Avrim Blum and Aaron Roth

Local Reconstructors and Tolerant Testers for Connectivity and Diameter 411
Andrea Campagna, Alan Guo, and Ronitt Rubinfeld

An Optimal Lower Bound for Monotonicity Testing over Hypergrids 425
Deeparnab Chakrabarty and C. Seshadhri

Small-Bias Sets for Nonabelian Groups: Derandomizations of the Alon-Roichman Theorem 436
Sixia Chen, Cristopher Moore, and Alexander Russell

What You Can Do with Coordinated Samples 452
Edith Cohen and Haim Kaplan

Robust Randomness Amplifiers: Upper and Lower Bounds 468
Matthew Coudron, Thomas Vidick, and Henry Yuen

The Power of Choice for Random Satisfiability 484
Varsha Dani, Josep Diaz, Thomas Hayes, and Cristopher Moore

Connectivity of Random High Dimensional Geometric Graphs 497
Roee David and Uriel Feige

Matching-Vector Families and LDCs over Large Modulo 513
Zeev Dvir and Guangda Hu

Explicit Noether Normalization for Simultaneous Conjugation via Polynomial Identity Testing 527
Michael A. Forbes and Amir Shpilka

Testing Membership in Counter Automaton Languages 543
Yonatan Goldhirsh and Michael Viderman

Tight Lower Bounds for Testing Linear Isomorphism 559
Elena Grigorescu, Karl Wimmer, and Ning Xie

Randomness-Efficient Curve Samplers 575
Zeyu Guo

Combinatorial Limitations of Average-Radius List Decoding 591
Venkatesan Guruswami and Srivatsan Narayanan

Zero Knowledge LTCs and Their Applications 607
Yuval Ishai, Amit Sahai, Michael Viderman, and Mor Weiss

A Tight Lower Bound for High Frequency Moment Estimation with Small Error 623
Yi Li and David P. Woodruff

Improved FPTAS for Multi-spin Systems 639
Pinyan Lu and Yitong Yin

Pseudorandomness for Regular Branching Programs via Fourier Analysis 655
Omer Reingold, Thomas Steinke, and Salil Vadhan

Absolutely Sound Testing of Lifted Codes 671
Elad Haramaty, Noga Ron-Zewi, and Madhu Sudan

On the Average Sensitivity and Density of k-CNF Formulas 683
Dominik Scheder and Li-Yang Tan

Improved Bounds on the Phase Transition for the Hard-Core Model in 2-Dimensions 699
Juan C. Vera, Eric Vigoda, and Linji Yang

Author Index 715

Spectral Sparsification in Dynamic Graph Streams

Kook Jin Ahn[1,⋆], Sudipto Guha[1,⋆], and Andrew McGregor[2,⋆⋆]

[1] University of Pennsylvania
{kookjin,sudipto}@seas.upenn.edu
[2] University of Massachusetts Amherst
mcgregor@cs.umass.edu

Abstract. We present a new bound relating edge connectivity in a simple, unweighted graph with effective resistance in the corresponding electrical network. The bound is tight. While we believe the bound is of independent interest, our work is motivated by the problem of constructing combinatorial and spectral sparsifiers of a graph, i.e., sparse, weighted sub-graphs that preserve cut information (in the case of combinatorial sparsifiers) and additional spectral information (in the case of spectral sparsifiers). Recent results by Fung et al. (STOC 2011) and Spielman and Srivastava (SICOMP 2011) show that sampling edges with probability based on edge-connectivity gives rise to a combinatorial sparsifier whereas sampling edges with probability based on effective resistance gives rise to a spectral sparsifier. Our result implies that by simply increasing the sampling probability by a $O(n^{2/3})$ factor in the combinatorial sparsifier construction, we also preserve the spectral properties of the graph. Combining this with the algorithms of Ahn et al. (SODA 2012, PODS 2012) gives rise to the first data stream algorithm for the construction of spectral sparsifiers in the dynamic setting where edges can be added or removed from the stream. This was posed as an open question by Kelner and Levin (STACS 2011).

1 Introduction

The main result of this paper is a bound between two basic graph quantities. First, let λ_e denote the edge-connectivity of edge $e = (s,t)$ in an unweighted graph G, i.e., the size of the minimum s-t cut. Second, consider the electrical network corresponding to G where every edge has unit resistance. Then, let r_e denote the effective resistance of edge $e = (s,t)$, i.e., the potential difference induced between s and t when a unit of current is injected at s and extracted at t. Then we show that $\lambda_e^{-1} \leq r_e \leq O(n^{2/3})\lambda_e^{-1}$.

Furthermore, there exist graphs where this inequality is tight. The first inequality is well known but the best existing upper bound for r_e in terms of λ_e

⋆ Research supported by NSF awards CCF-0644119, CCF-1117216 and a gift from Google.
⋆⋆ Research supported by NSF award CCF-0953754.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 1–10, 2013.

is $O(\sqrt{m})\lambda_e^{-1}$ where m is the number of edges in the graph [6]. Hence, our new bound is a strict improvement when $m = \Omega(n^{4/3})$. Indirectly related is work by Lyons et al. [17] that showed that for some classes of graphs, the effective resistance is within a constant factor of the corresponding edge connectivity if the graph is randomly weighted.

Graph Sparsification. The main idea in graph sparsification [4, 20] is to approximate a given graph $G = (V, E)$ by a sparse, weighted subgraph $H = (V, E', w)$. Throughout this paper, we will assume that G is unweighted and simple. A useful application of sparsifiers is in the design of faster algorithms for a range of problems including approximate max-flow [4] and sparsest cut [16]. The idea is that if H is a good approximation of G in an appropriate sense, then it suffices to solve the problem of interest on H rather than on G which would potentially have had many more edges. We say that H is a *combinatorial sparsifier* if

$$(1-\epsilon)\lambda_G(U) \leq \lambda_H(U) \leq (1+\epsilon)\lambda_G(U) \qquad \forall \text{ cuts } (U, V \setminus U) \tag{1}$$

where $\lambda_G(U)$ is the cardinality of the edges in E that cross the cut and $\lambda_H(U)$ is the total weight of the edges in E' that cross the cut. A more powerful sparsifier is a *spectral sparsifier* defined as follows. Let $L_G, L_H \in \mathbb{R}^{n\times n}$ by the Laplacian matrices of G and H respectively.[1] Then we say H is a spectral sparsifier of G if

$$(1-\epsilon)x^T L_G x \leq x^T L_H x \leq (1+\epsilon)x^T L_G x \qquad \forall x \in \mathbb{R}^n \ . \tag{2}$$

It is not hard to show that condition (2) implies condition (1) by relaxing the condition to only hold for $x \in \{0,1\}^n$.

It is perhaps surprising that, given any graph G, there exist sparsifiers for G with only $O(\epsilon^{-2} n \operatorname{polylog} n)$ edges. It is also possible to construct these subgraphs in near-linear time. In fact, construction is possible via the following two simple and elegant sampling algorithms:

1. *Combinatorial Sparsification* [11]: Sample each edge $e \in E$ independently with probability $p_e = \rho/\lambda_e$ where $\rho = \Theta(\epsilon^{-2}\log^2 n)$. Add each sampled edge e to H with weight $1/p_e$.
2. *Spectral Sparsification* [19]: Sample $\Omega(\epsilon^{-2} n \log n)$ edges with replacement where the probability p_e of picking e is proportional to r_e. Add each sampled edge e to H with weight s_e/p_e where s_e is the number of times e was sampled.

Roughly speaking, sampling e with respect to $1/\lambda_e$ is sufficient for preserving cut sizes whereas sampling e with respect to r_e also preserves additional spectral properties. In a sense, r_e is a more "nuanced" quantity since it takes into account not just the number of edge-disjoint paths between the end points of e but also the lengths of these paths. However, a consequence of our result is that if we simply over-sample by a factor $O(n^{2/3})$ in the above construction of a combinatorial sparsifier, we automatically preserve spectral information.

[1] Recall that $L_G = D_G - A_G$ where A_G is the adjacency matrix of G and D_G is the diagonal matrix where $D_G(i,i)$ is the degree of the ith node. Similarly $L_H = D_H - A_H$ where A_H is the weighted adjacency matrix of H and D_H is the diagonal matrix where $D_H(i,i) = \sum_{j:(i,j)\in E'} w_{ij}$ is the weighted degree of the ith node.

Dynamic Graph Streams and Sketches. The motivation for this work was the design and analysis of *graph sketching algorithms*, i.e., algorithms that use a (random) linear projection to compress a graph in such a way that relevant properties of the graph can be estimated from the projection with high accuracy and confidence. This use of linear projections is well-studied in the context of processing numerical data, e.g., signal reconstruction in compressed sensing [5, 8], Johnson-Lindenstrauss style dimensionality reduction [1, 12], and estimating properties of the frequency vectors that arise in data stream applications [7, 13]. However, it is only recently that it has been shown that this technique can be applied to more structured data such as graphs [2, 3].

Specifically, a sketch of a graph is defined as follows:

Definition 1 (Graph Sketches). *A* linear measurement *of a graph on n nodes is defined by a set of coefficients $\{c_e : e \in [n] \times [n]\}$. Given a graph $G = (V, E)$, the evaluation of this measurement is defined as $M_\lambda(G) = \sum_{e \in E} c_e$. A* sketch *is a collection of (non-adaptive) linear measurements. The size of this collection is referred to as the dimensionality of the sketch*

It was recently shown that there exist $O(\epsilon^{-2} n \operatorname{polylog} n)$-dimensional sketches from which a combinatorial sparsifier can be constructed [2,3]. This naturally gave rise to the first data stream algorithm for cut estimation when the stream is *fully-dynamic*, i.e., contains both edge insertions and deletions. The space-use of the algorithm is essentially the dimensionality of the sketch, i.e., $O(\epsilon^{-2} n \operatorname{polylog} n)$ and is therefore referred to as a *semi-streaming algorithm* [10]. This follows because each linear measurement can be evaluated on the stream using a single counter: on the insertion of e, we add c_e and on the deletion of e, we subtract c_e. In this paper, we develop the first data stream algorithm for constructing a spectral sparsifier. The algorithm uses $O(\epsilon^{-2} n^{5/3} \operatorname{polylog} n)$ space. In the case where there are no edge deletions, Kelner and Levin [15] designed an algorithm that used $O(\epsilon^{-2} n \operatorname{polylog} n)$ space. They posed the fully-dynamic case as an open problem.

2 Edge Connectivity vs. Effective Resistance

The proof is pleasantly simple and proceeds by considering the execution of the Edmonds-Karp algorithm [9], i.e., the Ford-Fulkerson algorithm that uses the shortest augmenting path first. Let d_e be the length of the $\lceil \lambda_e/2 \rceil$-th augmenting path when executing the algorithm.

We will use the following basic properties of effective resistances:

1. The resistance of a path is the sum of the resisters along the path.
2. The resistance of parallel paths is the harmonic mean of the resistances of the individual paths.
3. The resistance does not increase if edges are added to a graph.

Therefore the main idea would be to construct a suitable subgraph with the desired resistance and the result would follow.

Lemma 1. *The effective resistance on e is at most $2d_e\lambda_e^{-1}$.*

Proof. From the first $t = \lceil \lambda_e/2 \rceil$ augmenting paths, we can construct t edge-disjoint paths say $p_1, p_2, \ldots, p_t$. Note that the augmenting paths are directed and two augmenting paths can use the same original edge in both directions. Moreover the length of these directed flow augmenting paths is nondecreasing. As a consequence $\sum_i p_i \leq td_e$.

We now construct actual flow paths $\ell_1, \ell_2, \ldots, \ell_t$. In this process we elimininate cycles formed by using the edge in both directions. But as a result the length of a particular flow path can increase. However the cycle cancellation ensures that the total length is nonincreasing and $\sum_i \ell_i \leq \sum_i p_i$ which is sufficient for our proof here.

First assume these paths are vertex disjoint. If there were no other edges in the graph, r_e would be the harmonic mean of $\ell_1, \ldots, \ell_t$. However, adding extra edges will only decrease the effective resistance and hence

$$r_e \leq \frac{1}{\frac{1}{\ell_1} + \frac{1}{\ell_2} + \cdots + \frac{1}{\ell_t}} \leq \frac{\sum_i \ell_i}{t^2} \leq \frac{d_e}{t} \leq \frac{2d_e}{\lambda_e} \ .$$

where the second inequality is an application of the HM-AM inequality.

For the general case, we reduce to the vertex-disjoint case by removing all edges not in $\ell_1, \ldots, \ell_t$ to create circuit C_1 where any node v that is used in multiple paths is replaced by multiple copies (such that each path uses a distinct copy). Then, the effective resistance of e in the resulting circuit C_1 is at most the harmonic mean of $\ell_1, \ldots, \ell_t$ as before. Now consider adding 0 ohm resisters between each of the copies of an original node to give a new circuit C_2. Note that effective resistances in C_2 are less than in C_1 because C_2 was formed by adding edges. But then note that effective resistance in C_2 is the same as the effective resistance in the subgraph of G defined by these subset of edges because the voltage (potential) in each of the copies of an original node will be the same. Finally, adding additional edges to this subgraph does not increase the effective resistance. The lemma follows. □

Theorem 1. *In a simple, unweighted graph, the effective resistance on e is at most $O(n\lambda_e^{-3/2})$.*

Proof. Consider the residual graph after $\lceil \lambda_e/2 \rceil$ steps. The connectivity of e in the residual graph is $\lfloor \lambda_e/2 \rfloor$ and the shortest path length is at least d_e. It can then be shown[2] [14] that $d_e = O(n\lambda_e^{-1/2})$. Therefore, by appealing to Lemma 1, the effective resistance of e is at most $2d_e\lambda_e^{-1} \leq O(n\lambda_e^{-3/2})$. □

Corollary 1. *For simple, unweighted graphs, $\lambda_e^{-1} \leq r_e \leq O(n^{2/3})\lambda_e^{-1}$.*

[2] For completeness we include the argument here. Let $e = (s,t)$ and let $\Gamma_i(s)$ be the set of nodes with distance exactly i from s. Because there are still at least $\lfloor \lambda_e/2 \rfloor$ edge disjoint paths between s and t we know $(|\Gamma_i(s)| + |\Gamma_{i+1}(s)|)^2 \geq \lfloor \lambda_e/2 \rfloor$ for each i. Hence, $\Omega(d_e\sqrt{\lambda_e}) = \sum_{i=1}^{d_e} |\Gamma_i(s)| \leq n$ and therefore $d_e = O(n\lambda_e^{-1/2})$ as required.

Proof. First, note that $r_e \leq 1$ and so if $n\lambda_e^{-3/2} \geq 1$, we have

$$r_e \leq 1 \leq (n\lambda_e^{-3/2})^{2/3} = n^{2/3}\lambda_e^{-1} ,$$

as required. On the other hand, if $n\lambda_e^{-3/2} \leq 1$, then by Theorem 1 we also have

$$r_e \leq O\left(n\lambda_e^{-3/2}\right) \leq O\left(n^{2/3}\lambda_e^{-1}\right).$$

□

2.1 A Tight Example

We next show that the bound in Corollary 1 is tight. Consider a layered graph with layers $L_0 = \{s\}, L_1, \ldots, L_k, L_{k+1} = \{t\}$ with edges

$$E = \{(u, v) : u \in L_i, v \in L_{i+1} \text{ for some } i \in \{0, 1, 2, \ldots, k\}\} \cup \{(s, t)\} .$$

Let $e = (s, t)$. Let $k = n/\sqrt{\lambda_e}$. Setting $|L_1| = |L_k| = \lambda_e - 1$ and $|L_1| = |L_2| = \ldots = |L_{k-1}| = \sqrt{\lambda_e - 1}$ ensures that the minimum $s - t$ cut has size λ_e. The total number of nodes is $\Theta(n)$ on the assumption that $\lambda_e = O(n)$. The graph is illustrated in Figure 1.

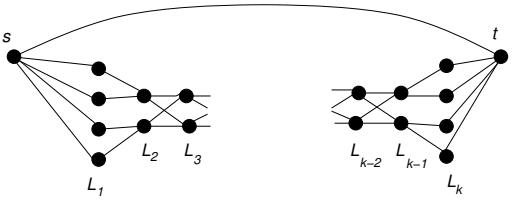


Fig. 1. Tight example for Corollary 1

The above network, when considered from the perspective of s and t is the same as the following network in Figure 2; the nodes in $L_1, \ldots, L_k$ all have the same potential and the capacity (parallel edges) between adjacent nodes is $\lambda_e - 1$. Therefore, using the fact that for parallel edges the effective resistance is the harmonic mean of the resistances, the effective resistance between L_i, L_{i+1} is $1/(\lambda_e - 1)$ when we ignore edge e. It follows that, if we ignore the edge (s, t), then the effective resistance between s and t is $\Theta(k\lambda_e^{-1}) = \Theta(n\lambda_e^{-1.5})$ since resistances connected in series are additive. If we set $\lambda_e = n^{2/3}$ then $n\lambda_e^{-1.5} = 1$ and consequently $r_e = \Theta(1)$. Therefore, the edge $e = (s, t)$ satisfies $r_e = \Theta(n^{2/3}/\lambda_e)$ and hence Corollary 1 is tight.

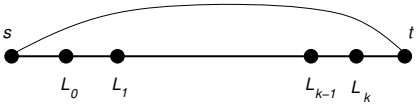


Fig. 2. Network Equivalent to Figure 1 (from the perspective of s, t)

3 Spectral Sparsification

Our sparsification result uses the following theorem[3] due to Spielman and Srivastava [19].

Theorem 2. *Given a graph G, let $\{z_e\}_{e\in E}$ be a set of positive values that satisfy:*

1. *$z_e \geq r_e$ for all $e \in E$*
2. *$\sum_e z_e = \beta \sum_e r_e$ for some $\beta \geq 1$*

Sample $q \geq c_0\beta\epsilon^{-2}n\log n$ edges with replacement where edge e is chosen with probability $p_e = z_e/\sum_e z_e$ where c_0 is an absolute constant. Let H be the weighted graph where edge e has weight $s_e/(qp_e)$ where s_e is the number of times e was sampled. Then,

$$(1-\epsilon)x^T L_G x \leq x^T L_H x \leq (1+\epsilon)x^T L_G x \qquad \forall x \in \mathbb{R}^n \ ,$$

with probability at least $1/2$.

In what follows z_e will take a known value between α/λ_e and $4\alpha/\lambda_e$ where $\alpha = \Theta(n^{2/3})$ is chosen according to Corollary 1 such that $\alpha/\lambda_e \geq r_e$. Since $r_e \geq 1/\lambda_e$, we have

$$\sum_e z_e \leq \sum_e 4\alpha/\lambda_e \leq 4\alpha \sum_e r_e$$

as required. Therefore, to construct a spectral sparsifier of a dynamic graph stream, it suffices to emulate that above sampling procedure. To do this, we divide the procedure into the following two steps:

1. *Independent Edge Sampling:* We first show how to sample edges such that each edge is sampled independently and e is sampled with probability $y_e = \min\{8c_0 z_e \epsilon^{-2}\log n, 1\}$. This will be performed in a single-pass over a dynamic graph stream using $O(\epsilon^{-2}n^{5/3}\operatorname{polylog} n)$ space. Let S be the set of samples returned.
2. *Emulate Sampling with Replacement:* Given the sampled edges S along with $\{z_e\}_{e\in S}$ and $Z = \sum_e z_e$, we show that it is possible to emulate the required sampling with replacement. This will be performed in post-processing without requiring an additional pass over the data stream.

In the next section we show how to perform the independent edge sampling with the required parameters. However, emulating sampling replacement is straightforward and is described in the proof of the next lemma.

Lemma 2. *Let $S = \{e_1, e_2, \ldots\}$ be the edges returned by the independent edge sampling and let $\{z_e\}_{e\in E}$ be the sampling parameters with $Z = \sum_{e\in E} z_e$. Then, given a positive integer $q \in [c_0\beta\epsilon^{-2}n\log n, 2c_0\beta\epsilon^{-2}n\log n]$ suppose that we sample each edge e with probability at least $y_e = \min\{8c_0 z_e\epsilon^{-2}\log n, 1\}$, then after the independent edge sampling is done, we can emulate the process of sampling q edges with replacement such that each edge is chosen with probability proportional to z_e.*

[3] The theorem follows from Corollary 6 in [19] after a re-parameterization of ϵ and by scaling z_e such that we ensure $z_e \geq r_e$.

Proof. Consider a Poisson process with parameter z_e for each edge e and collect the first q edges. This sampling process is equivalent to sample q edges with replacement where the probability of sampling an edge e is z_e/Z. Recall $Z = \beta \sum_e r_e = \beta(n-1)$ since $\sum_e r_e = n-1$ for any graph [6,19].

We use the independent sampling to emulate this process with high probability. Then, the expected number of edges sampled in a unit time in the above Poisson process is Z and therefore, we sample q edges in $2q/Z$ time with high probability assuming that q is sufficiently large (note that $q \geq \Omega(n\epsilon^{-2}\log n)$).

Let t_e be the random variable which indicates the first time when e appears in the above Poisson process. If $t_e > 2q/r$, we can safely ignore e because then e is not going to be sampled with high probability. Otherwise, we want to store e to emulate the process. By the definition, $t_e \sim \mathrm{Exp}(z_e)$ and we have

$$\mathbb{P}\left[t_e \leq \frac{2q}{Z}\right] = 1 - \exp\left(-z_e \frac{2q}{Z}\right) \leq \frac{2qz_e}{Z} \leq \frac{4c_0\beta n\epsilon^{-2}\log n}{\beta(n-1)} z_e \leq 8c_0 z_e \epsilon^{-2}\log n$$

Therefore, if we sample e with the stated probability or higher, we can emulate the Poisson process for e upto time $2q/Z$ which is equivalent to emulate the sampling with replacement with high probability. □

3.1 Independent Edge Sampling

Our sampling makes use of the following two existing algorithms for dynamic graph streams [2,3]:

1. k-EDGE-CONNECT: Given a dynamic graph stream defining a graph G, this algorithm returns a subgraph $S = k\text{-EDGE-CONNECT}(G)$ such that with high probability

$$\lambda_S(U) \geq \min(k, \lambda_G(U)) \qquad \text{for any cut } (U, V \setminus U)$$

   The algorithm uses $O(kn\,\mathrm{polylog}\,n)$ bits of space.
2. CUT-SPARSIFIER: Given a dynamic graph stream defining a graph G, this algorithm returns a weighted subgraph $H = \text{CUT-SPARSIFIER}(G)$ such that with high probability

$$\lambda_G(U) \leq \lambda_H(U) \leq 2\lambda_G(U) \qquad \text{for any cut } (U, V \setminus U)$$

   The algorithm uses $O(n\,\mathrm{polylog}\,n)$ bits of space.

The idea behind our sampling algorithm is to subsample the edges of the graph at $O(\log n)$ different rates, $1/2, 1/4, 1/8, \ldots$. At each sampling rate, or level, we maintain a "skeleton" that ensures that we have at least a certain number of edges across each cut. We then return an edge e if it appears in the skeleton at a particular level where the level should be chosen proportional to $1/\lambda_e$. We use CUT-SPARSIFIER as an oracle that can provide estimates for every λ_e value in post-processing. Specifically, the new edge-sampling algorithm operates as follows:

1. During a single pass of the stream:
   - Construct $H = \text{CUT-SPARSIFIER}(G)$.
   - Construct $T_i = k\text{-EDGE-CONNECT}(G_i^z)$ for $i = 0, 1, 2, \ldots, 2\lg n$ where $G_i^z = (V, E_i^z)$ is the graph formed by sampling each edge in G with probability 2^{-i} and k is set to $16 \ln n$.
   - Construct $S_i = k\text{-EDGE-CONNECT}(G_i)$ for $i = 0, 1, 2, \ldots, 2\lg n$ where $G_i = (V, E_i)$ is the graph formed by sampling each edge in G with probability 2^{-i} and k is set to $64 c_0 \alpha \epsilon^{-2} \log n$.
2. Post-Processing:
   - From H, let $\tilde{\lambda}_e$ be an estimate of λ_e and note that $\lambda_e \leq \tilde{\lambda}_e \leq 2\lambda_e$ by the guarantee of the CUT-SPARSIFIER algorithm.
   - Using $F = \{e : e \in T_i \text{ where } i = \max(0, \left\lfloor \lg(\tilde{\lambda}_e / \ln n) \right\rfloor - 1)\}$, estimate $Z = \sum_e z_e$:
     (a) For $e \in F$, compute $f_e = 2^{\min(0, -\lfloor \lg(\tilde{\lambda}_e / \ln n) \rfloor + 1)}$ and $z_e = 2\alpha/\tilde{\lambda}_e$.
     (b) Return $\tilde{Z} = \sum_{e \in F} z_e / f_e$ as an estimation of Z (See Lemma 4).
   - Let $z_e = 2\alpha/\tilde{\lambda}_e$ and $y_e = \min\{8 c_0 z_e \epsilon^{-2} \log n, 1\}$. Note that,

$$z_e \leq \frac{2\alpha}{\lambda_e} \leq 2\alpha r_e \qquad \text{and} \qquad z_e = \frac{2\alpha}{\tilde{\lambda}_e} \geq \frac{2\alpha}{2\lambda_e} = \frac{\alpha}{\lambda_e} \geq r_e \ ,$$

     which satisfies the desiderata of Theorem 2.
   - Return $\{e : e \in S_i \text{ where } i = \lfloor \lg 1/y_e \rfloor\}$ as the required set of samples for Lemma 2. We now have a set of independent samples with replacement which satisfies Theorem 2 and the sparsifier can be constructed as stated therein. This is proved in Lemma 3.

Note that, for the sake of analysis, we assume each edge is included in G_i independently. However, to implement that algorithm in small space we would actually use Nisan's pseudo-random generator [18]. While this is not necessarily the most efficient way to generate the random variables (it adds additional logarithmic terms to the running time and space complexity), this approach leads to a simpler description of the algorithm.

Lemma 3. *For all $e \in E$ and $i = \lfloor \lg 1/y_e \rfloor$, the edge e is sampled in G_i with probability between y_e and $2y_e$. With high probability, $e \in S_i$ iff $e \in G_i$.*

Proof. Clearly $e \in S_i$ implies $e \in G_i$ since S_i is a subgraph of G_i. For the other direction, assume $e = (s, t) \in G_i$ and let $E_e \subseteq E$ be the edges across the minimum s-t cut in G. In particular, $|E_e| = \lambda_e$. But

$$\mathbb{E}\left[|E_e \cap E_i|\right] \leq 32 c_0 \alpha \epsilon^{-2} \log n$$

and by an application of the Chernoff bound, $|E_e \cap E_i| < 64 c_0 \alpha \epsilon^{-2} \log n$ with high probability. Hence $e \in S_i$ for $k = 64 c_0 \alpha \epsilon^{-2} \log n$ by appealing to the guarantees of the k-EDGE-CONNECT algorithm. By an application of the union bound, this is true for all $e \in E$ with high probability as well. □

The next lemma establishes that we also have a good estimate of $Z = \sum_e z_e$.

Lemma 4. *With high probability,* $(1-\epsilon)Z \leq \tilde{Z} \leq (1+\epsilon)Z$.

Proof. Using an argument almost identical to the proof of Lemma 3 we argue that the algorithm (with high probability) samples each edge e in F with probability f_e. Since f_e is the probability that $e \in F$,

$$\mathbb{E}\left[\tilde{Z}\right] = \mathbb{E}\left[\sum_{e \in F} z_e/f_e\right] = \sum_{e \in E} z_e \ .$$

In addition, $z_e/f_e \leq \epsilon^2 Z/\log n$. By an application of the Chernoff bound, $(1-\epsilon)Z \leq \tilde{Z} \leq (1+\epsilon)Z$ with high probability. □

4 Conclusions

While the bound we establish relating λ_e and r_e is tight up to constant factors, the resulting data stream algorithm need not be optimal. Specifically, we conjecture that spectral sparsification is possible in the semi-streaming model where the algorithm may use $O(n \operatorname{polylog} n)$ space.

References

1. Achlioptas, D.: Database-friendly random projections: Johnson-lindenstrauss with binary coins. J. Comput. Syst. Sci. 66(4), 671–687 (2003)
2. Ahn, K.J., Guha, S., McGregor, A.: Analyzing graph structure via linear measurements. In: SODA, pp. 459–467 (2012)
3. Ahn, K.J., Guha, S., McGregor, A.: Graph sketches: sparsification, spanners, and subgraphs. In: PODS, pp. 5–14 (2012)
4. Benczúr, A.A., Karger, D.R.: Approximating s-t minimum cuts in $\tilde{o}(n^2)$ time. In: STOC, pp. 47–55 (1996)
5. Candès, E.J., Romberg, J.K., Tao, T.: Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Transactions on Information Theory 52(2), 489–509 (2006)
6. Christiano, P., Kelner, J.A., Madry, A., Spielman, D.A., Teng, S.-H.: Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In: STOC, pp. 273–282 (2011)
7. Cormode, G.: Sketch techniques for approximate query processing. In: Cormode, G., Garofalakis, M., Haas, P., Jermaine, C. (eds.) Synposes for Approximate Query Processing: Samples, Histograms, Wavelets and Sketches. Foundations and Trends in Databases. NOW Publishers (2011)
8. Donoho, D.L.: Compressed sensing. IEEE Transactions on Information Theory 52(4), 1289–1306 (2006)
9. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) Combinatorial Optimization (Edmonds Festschrift). LNCS, vol. 2570, pp. 31–33. Springer, Heidelberg (2003)
10. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. Theor. Comput. Sci. 348(2-3), 207–216 (2005)

11. Fung, W.S., Hariharan, R., Harvey, N.J.A., Panigrahi, D.: A general framework for graph sparsification. In: STOC, pp. 71–80 (2011)
12. Johnson, W.B., Lindenstrauss, J.: Extensions of Lipshitz mapping into Hilbert Space. Contemporary Mathematics 26, 189–206 (1984)
13. Kane, D.M., Nelson, J., Woodruff, D.P.: An optimal algorithm for the distinct elements problem. In: PODS, pp. 41–52 (2010)
14. Karzanov, A.: Determining a maximal flow in a network by the method of preflows. Soviet Math. Dokl. 15(2) (1974)
15. Kelner, J.A., Levin, A.: Spectral sparsification in the semi-streaming setting. In: STACS, pp. 440–451 (2011)
16. Khandekar, R., Rao, S., Vazirani, U.V.: Graph partitioning using single commodity flows. J. ACM 56(4) (2009)
17. Lyons, R., Pemantle, R., Peres, Y.: Resistance bounds for first-passage percolation and maximum flow. J. Comb. Theory, Ser. A 86(1), 158–168 (1999)
18. Nisan, N.: Pseudorandom generators for space-bounded computation. Combinatorica 12(4), 449–461 (1992)
19. Spielman, D.A., Srivastava, N.: Graph sparsification by effective resistances. SIAM J. Comput. 40(6), 1913–1926 (2011)
20. Spielman, D.A., Teng, S.-H.: Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In: STOC, pp. 81–90 (2004)

The Online Stochastic Generalized Assignment Problem

Saeed Alaei *, MohammadTaghi Hajiaghayi *, and Vahid Liaghat *

Department of Computer Science, University of Maryland,
College Park, MD 20742
{saeed,hajiagha,vliaghat}@cs.umd.edu

Abstract. We present a $1 - \frac{1}{\sqrt{k}}$-competitive algorithm for the online stochastic generalized assignment problem under the assumption that no item takes up more than $\frac{1}{k}$ fraction of the capacity of any bin. Items arrive online; each item has a value and a size; upon arrival, an item can be placed in a bin or discarded; the objective is to maximize the total value of the placement. Both value and size of an item may depend on the bin in which the item is placed; the size of an item is revealed only after it has been placed in a bin; distribution information is available about the value and size of each item in advance (not necessarily i.i.d), however items arrive in adversarial order (non-adaptive adversary).

We also present an application of our result to subscription-based advertising where each advertiser, if served, requires a given minimum number of impressions (i.e., the *"all or nothing"* model).

1 Introduction

The generalized assignment problem (GAP) and its special cases *multiple knapsack*[1] and *bin packing*[2] capture several fundamental optimization problems and have many practical applications in computer science, operations research, and related disciplines. The (offline) GAP is defined as follows:

Definition 1 (Generalized Assignment Problem). *There is a set of n items and m bins. Each bin has a hard capacity where the total size of items placed in a bin cannot exceed its capacity. Each item has a value and a size if placed in a bin; both might depend on the bin. The goal is to find a maximum valued assignment of items to the bins which respects the capacities of the bins.*

For example GAP can be viewed as a scheduling problem on parallel machines, where each machine has a capacity (or a maximum load) and each job has a size (or a processing time) and a profit, each possibly dependent on the machine to which it is assigned, and the objective is to find a feasible scheduling which maximizes the total

* Supported in part by NSF CAREER award 1053605, ONR YIP award N000141110662, DARPA/AFRL award FA8650-11-1-7162, a Google Faculty Research Award, and a University of Maryland Research and Scholarship Award (RASA).

[1] In the multiple knapsack problem, we are given a set of items and a set of bins (knapsacks) such that each item j has a profit v_j and a size s_j, and each bin i has a capacity C_i. The goal is to find a subset of items of maximum profit such that they have a feasible packing in the bins.

[2] In the bin packing problem, given a set of items with different sizes, the goal is to find a packing of items into unit-sized bins that minimizes the number of bins used.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 11–25, 2013.

profit. Though multiple knapsack and bin packing have a fully polynomial-time approximation scheme (asymptotic for bin packing) in the offline setting [6], GAP is APX-hard and the best known approximation ratio is $1 - 1/e + \epsilon$ where $\epsilon \approx 10^{-180}$ [10], which improves on a previous $(1 - 1/e)$-approximation [12].

In this paper we consider the *online stochastic* variant of the problem:

Definition 2 (Online Stochastic Generalized Assignment Problem). *There are n items arriving in an online manner which can be of different types. There are m (static) bins each with a capacity limit on the total size of items that can be placed in it. A type of an item is associated with a value and a size distribution which may depend on the bin to which the item is placed. Stochastic information is known about the type of an item and sizes/values of the types. Upon arrival of an item, the type of that item is revealed. However the realization of the size of the item is revealed only after it has been placed in a bin. The goal is to find a maximum valued assignment of items to the bins. We consider a* large-capacity *assumption: no item takes up more than $\frac{1}{k}$ fraction of the capacity of any bin.*

We emphasize that there are two sources of uncertainty in our model: the type of an item and the size of the item. The type of an item (which contains a size-distribution) is revealed before making the assignment, however the actual size of an item is revealed after the assignment. To the best of our knowledge, Feldman et al. [11] were the first to consider the generalized assignment problem in an online setting, albeit with deterministic sizes. In the adversarial model where the items and the order of arrivals are chosen by an adversary, there is no competitive algorithm. Consider the simple case of one bin with capacity one and two arriving items each with size one. The value of the first item is 1. The value of the second item would be either $\frac{1}{\epsilon}$ or 0 based on whether we place the first item in the bin. Thus the online profit cannot be more than ϵ factor of the offline profit. Indeed one can show a much stronger hardness result for the adversarial model: no algorithm can be competitive for the two special cases of GAP, namely the adword problem[3] and the display ad problem[4] even under the large-capacity assumption [11,19].

Since no algorithm is competitive for online GAP in the adversarial model, Feldman et al. consider this model with *free disposal*. In the free disposal model, the total size of items placed in a bin may exceed its capacity, however, the profit of the bin is the maximum-valued subset of the items in the bin which does not violate the capacity. Feldman et al. give a $(1 - \frac{1}{e} - \epsilon)$-competitive primal-dual algorithm for GAP under the free disposal assumption and the additional large-capacity assumption by which the capacity of each bin is at least $O(\frac{1}{\epsilon})$ times larger than the maximum size of a single item. Although the free disposal assumption might be counter-intuitive in time-sensitive applications such as job scheduling where the machine may start doing a job right after the job assignment, it is a very natural assumption in many applications including applications in economics like ad allocation – a buyer does not mind receiving more items.

[3] The adword problem is a special case of GAP where the size and the value of placing an item in a bin is the same.

[4] The display ad problem is a special case of GAP where all sizes are uniform.

Dean, Goemans, and Vondrak [7] consider the (offline) stochastic knapsack problem which is closely related to GAP. In their model, there is only one bin and the value of each item is known. However, the size of each item is drawn from a known distribution only after it is placed in the knapsack. We note that this is an offline setting in the sense that we may choose any order of items for allocation. This model is motivated by job scheduling on a single machine where the actual processing time required for a job is learned only after the completion of the job. Dean et al. give various adaptive and non-adaptive algorithms for their model where the best one has a competitive ratio $\frac{1}{3} - \epsilon$. Recently Bhalgat improved the competitive ratio to $\frac{1}{2} - \epsilon$ [4]. Other variations, such as soft capacity constraints, have also been considered for which we refer the reader to [5,13,16]. Dean et al. [7] also introduce an *ordered model* where items must be considered in a specific order, which can be seen as a version of the the online model with a known order. Dean et al. [7] present a $\frac{1}{9.5}$-competitive algorithm. In general, the online model can be considered as a more challenging variation of the models proposed by Dean et al, and we show that the large-capacity assumption is enough to overcome this challenge.

To the best of our knowledge, the current variation of the online stochastic GAP has not been considered before. We note that since the distributions are not necessary i.i.d., this model generalizes the well-known *prophet inequalities*. [5] Even with stochastic information about the arriving queries, no online algorithm can achieve a competitive ratio better than $\frac{1}{2}$ [1,14,17,18]. Consider the simple example from before where the value of the first item is 1 with probability one and the value of the second item is $\frac{1}{\epsilon}$ with probability ϵ, and 0 with probability $1-\epsilon$. The algorithm can only select one item. No online (randomized) algorithm can achieve a profit more than $\max\{1, \epsilon(\frac{1}{\epsilon})\} = 1$ in expectation. However, the expected profit of the optimum offline assignment is $(1-\epsilon)1 + \epsilon(\frac{1}{\epsilon}) = 2 - \epsilon$. Therefore without any additional assumption one cannot get a competitive ratio better than $1/2$. We overcome this difficulty by considering the natural large-capacity assumption which arises in many applications such as online advertising.

A summary of the other related work is presented in the full version of the paper.

1.1 Our Contribution

The *loss factor* of an online algorithm is the ratio α such that the profit of the algorithm is at least $1 - \alpha$ fraction of the optimal offline profit. The main result of the paper can be summarized in the following theorem (formally stated in Theorem 4).

Theorem. *For the stochastic generalized assignment problem, there exists a randomized online algorithm (see Definition 6) with the loss factor at most $\frac{1}{\sqrt{k}}$ in expectation.*

The proposed algorithm initially computes an optimal solution for a linear program corresponding to a fractional expected instance. In the online stage, the algorithm tentatively assigns each item upon arrival to one of the bins at random with probabilities

[5] In the classic prophet inequality problem, given the distribution of a sequence of random variables, an onlooker has to choose from the succession of these values. The onlooker can only choose a certain number of values and cannot choose a past value. The goal is to maximize the total sum of selected numbers.

proportional to the fractional LP solution. This ensures that the expected total size of items assigned tentatively to each bin does not exceed its capacity. However, once a bin becomes full, any item which gets tentatively assigned to that bin will have to be discarded. In general, a straightforward randomized assignment based on the LP solution could be arbitrarily far from optimal; that is because the probability of an item being discarded due to a bin being full could be arbitrarily close to 1 for items that arrive towards the end. To overcome this problem, we incorporate an adaptive threshold based strategy for each bin so that an item tentatively assigned to a bin is only placed in the bin if the remaining capacity of the bin is more than a certain threshold. This ensures the online algorithm discards a tentatively assigned item with a probability at most $\frac{1}{\sqrt{k}}$ of the fractional LP assignment. The thresholds are computed adaptively based on previously observed items.

Indeed by using the fractional solution as a guideline, it is possible to achieve a non-adaptive competitive algorithm. One may scale down the fractional solution by a factor $1 - O(\frac{\log k}{\sqrt{k}})$ and assign the items with the modified probabilities, thus achieving a loss factor $O(\frac{\log k}{\sqrt{k}})$. By using the Chernoff bound it can be shown that the probability of exceeding bin capacities is very small. This simple algorithm gives an asymptotically optimum solution, however there are two drawbacks. The first issue is that the constants in various implementations of this idea are large. Thus unless the value k is very large, this algorithm cannot guarantee a reasonable competitive ratio; in contrast, the loss factor of our algorithm is exactly $\frac{1}{\sqrt{k}}$. On the other hand, in the applications of online GAP such as the Adword problem, the factor $O(\frac{\log k}{\sqrt{k}})$ is the loss factor of the millions of dollars. Therefore our algorithm saves a logarithmic factor in the loss of revenue. Indeed these drawbacks were also the motivation for improving the loss factor in the special cases of online GAP in previous papers [2,3].

The threshold based strategy of the online algorithm is presented in Section 3 in the form of a generalization of the magician's problem of [1]. The original magician's problem can be interpreted as a stochastic knapsack problem with unit size items and a knapsack of size k and such that each item arrives in one of two possible states (e.g. good/bad) with known probabilities; the objective being to maximize – simultaneously for all items – the probability of picking every item that is good. On the other hand, in the generalized variant presented in the current paper, the size of each item can vary according to an arbitrary (but known) distribution; in this version k is an integer lower bound on the ratio of the total size of the knapsack to the maximum possible size of a single item. Although the bound we obtain for the generalized magician is similar to that of [1], they are incomparable for small k; in particular for $k = 1$, one can easily achieve a $\frac{1}{2}$-competitive algorithm for the magician's problem with fixed size items, whereas for the generalized version with variable size items, no constant competitive algorithm is possible for $k = 1$.

Recently, Alaei et al. [3,2][6] use a combination of expected linear programming approach and dynamic programming to achieve a $1 - \epsilon$-competitive algorithm for adword and display ad. They use a relatively simple dynamic programming in combination with

[6] In an independent work, Devanur et al.[8] also consider the expected linear program of a similar problem.

the LP solution to check whether they should assign an item to a bidder or discard it. Using an approach similar to "dual fitting" [15], they demonstrate an analysis of the combination of a dynamic programming approach with an online LP-based algorithm and prove a loss factor $\frac{1}{\sqrt{k+3}}$ for the display ad problem. They use the sand theorem of [1] as a black-box in their analysis to derive proper dual variables in their dual fitting analysis of the algorithm. A dynamic programming approach needs to know the stochastic information about the remaining items, while a threshold-based approach needs to know the past, i.e., they are complements of each other. However, analysis of the dynamic programming even with uniform sizes is involved. Furthermore, it is not easy to generalize the approach of [2] to the model of Goemans et al. [7] where the given sizes show only the expected size of an item.

It is worthwhile to compare the stochastic model of the current paper against other popular models, i.e., random order model[7] and unknown distribution model[8]. While both the random order and the unknown distribution models require less stochastic information, they both treat items uniformly; hence they are more suitable for applications where items are more symmetric[9]. The model considered in the current paper is more suitable when there is a high degree of distribution asymmetry across the items. In particular, the extra stochastic information allows us to obtain practical bounds even for small values of k whereas in other stochastic models the obtained bounds often become meaningful only asymptotically in k.

The all-or-nothing model. The online algorithm of this paper can be applied to a slightly different model in which each item should still be either fully assigned or discarded, however in case of assignment, unlike GAP, an item can be fractionally split across multiple bins (i.e., the all-or-nothing assignment model). Note that the LP for the expected instance is still the same for the all-or-nothing model, therefore our online algorithm still obtains the same bound in expectation compared to the optimal offline solution. The all-or-nothing model is suitable for *subscription-based advertisement* and *banner advertisement*.

The subscription-based advertisement problem is an example of an offline ad allocation setting motivated by the banner advertisement. In this problem, there is a set of contracts proposed by the advertisers and the goal is to accept the contracts of a subset of the advertisers which maximizes the revenue. The contract proposed by an advertiser specifies a collection of webpages which are relevant to his product, a desired total quantity of impressions on these pages, and a price. Each webpage has an ad inventory with a certain number of banner ads. The problem of selecting a feasible subset of advertisers with the maximum total value does not have any non-trivial approximation. This can be shown by a reduction from the Independent Set problem on a graph; advertisers represent the vertices of the graph and webpages represent the edges of the graph. Advertisers desire all the impressions of the relevant webpages. Thus any feasible subset of advertisers would denote an independent set in the graph. This shows maximizing

[7] Random arrival model: items are chosen by an adversary, but they arrive in a random order.

[8] Unknown distribution model: items are chosen i.i.d. from a fixed but unknown distribution.

[9] At a first glance, the random order model may appear to allow for asymmetry, however note that for any i and j, the i^{th} arriving item and the j^{th} arriving item have the same ex ante distribution in the random order model.

the total value does not have a non-trivial approximation. Different pricing models have also been introduced by Feige et al. [9]. The proof of the following corollary is by a reduction from the all-or-nothing model.

Corollary. *There exists a randomized algorithm for the subscription-based advertisement problem which obtains a loss factor $\frac{1}{\sqrt{k}}$ in expectation where the number of available impressions on each website is at least k times the required impressions of each relevant advertiser.*

Proof. One can show that this is an offline version of the all-or-nothing model where bins denote the webpages and items denote the advertisers. The size of an item is the required number of ads of an advertiser and the value of an item is the proposed price. By Theorem 4, we can achieve at least $1 - \frac{1}{\sqrt{k}}$ fraction of the optimal profit in expectation.

2 Preliminaries

Model. We consider the problem of assigning n items to m bins; items arrive online and stochastic information is known about the size/value of each item; the objective is to maximize the total value of the assignment. The item $i \in [n]$ (arriving at time i) has r_i possible types with each type $t \in [r_i]$ having a probability of p_{it}, a value of $v_{itj} \in \mathbb{R}_+$, and a size of $S_{itj} \in [0,1]$ if placed in bin j (for each $j \in [m]$); S_{itj} is a random variable which is drawn from a distribution with a CDF of F_{itj} if the item is placed in bin j. Each bin $j \in [n]$ has a capacity of $c_j \in \mathbb{N}_0$ which limits the total size of the items placed in that bin[10]. The type of each item is revealed upon arrival and the item must be either placed in a bin or discarded; this decision cannot be changed later. The size of an item is revealed only after it has been placed in a bin, furthermore an item can be placed in a bin only if the bin has at least one unit of capacity left. We assume that n, m, c_j, v_{itj} and F_{itj} are known in advance.

Benchmark. Consider the following linear program in which $\widetilde{s}_{itj} = \mathbf{E}_{S_{itj} \sim F_{itj}}[S_{itj}]$ [11].

$$\begin{aligned}
\text{maximize} \quad & \sum_i \sum_t \sum_j v_{itj} x_{itj} && && (\overline{OPT}) \\
\text{subject to} \quad & \sum_i \sum_t \widetilde{s}_{itj} x_{itj} \le c_j, && \forall j \in [m] \\
& \sum_j x_{itj} \le p_{it}, && \forall i \in [n], \forall t \in [r_i] \\
& x_{itj} \in [0,1],
\end{aligned}$$

The optimal value of this linear program, which corresponds to the expected instance, is an upper bound on the expected value of the optimal offline assignment.

[10] Our results hold for non-integer capacitates, however we assume integer capacities to simplify the exposition.

[11] Throughout the rest of this paper, we often omit the range of the sums whenever the range is clear from the context (e.g., $\sum_i$ means $\sum_{i \in [n]}$, and $\sum_j$ means $\sum_{j \in [m]}$, etc).

Theorem 1. *The optimal value of the linear program ($\overline{OPT}$) is an upper bound on the expected value of the offline optimal assignment.*

Proof. Let x^*_{itj} denote the ex ante probability that item i is of type t and is assigned to bin j in the optimal offline assignment. It is easy to see that x^*_{itj} is a feasible assignment for the linear program. Furthermore, the expected value of the optimal offline assignment is exactly $\sum_i \sum_t \sum_j v_{itj} x^*_{itj}$ which is equal to the value of the linear program for x^*_{itj} which is itself no more than the optimal value of the linear program. Note that the optimal value of the linear program may be strictly higher since a feasible assignment of the linear program does not necessarily correspond to a feasible offline assignment policy.

Section 4 presents an online adaptive algorithm which obtains a loss factor $\frac{1}{\sqrt{k}}$ w.r.t. the optimal value of the above linear program, where $k = \min_j c_j$. We emphasize that our adaptive algorithm saves a logarithmic factor in the loss of the outcome compared to the non-adaptive methods. Next section presents a stochastic toy problem and its solution which is used in our online algorithm.

3 The Generalized Magician's Problem

We present a generalization of the magician's problem, which was originally introduced in [1]; we also present a near-optimal solution for this generalization.

Definition 3 (The Generalized Magician's Problem). *A magician is presented with a series of boxes one by one, in an online fashion. There is a prize hidden in one of the boxes. The magician has a magic wand that can be used to open the boxes. The wand has k units of mana [20]. If the wand is used on box i and has at least 1 unit of mana, the box opens, but the wand loses a random amount of mana $X_i \in [0, 1]$ drawn from a distribution specified on the box by its cumulative distribution function F_{X_i} (i.e., the magician learns F_{X_i} upon seeing box i). The magician wishes to maximize the probability of obtaining the prize, but unfortunately the sequence of boxes, the distributions written on the boxes, and the box containing the prize have been arranged by a villain; the magician has no prior information (not even the number of the boxes); however, it is guaranteed that $\sum_i \mathbf{E}[X_i] \leq k$, and that the villain has to prepare the sequence of boxes in advance (i.e., cannot make any changes once the process has started).*

The magician's problem introduced in [1] is a special case of this model where $X_i = 1$ for every i.

The magician could fail to open a box either because (a) he might choose to skip the box, or (b) his wand might run out of mana before getting to the box. Note that once the magician fixes his strategy, the best strategy for the villain is to put the prize in the box which, based on the magician's strategy, has the lowest ex ante probability of being opened. Therefore, in order for the magician to obtain the prize with a probability of at least γ, he has to devise a strategy that guarantees an ex ante probability of at least γ for opening each box. Notice that allowing the prize to be split among multiple boxes does not affect the problem. We present an algorithm parameterized by a probability

$\gamma \in [0,1]$ which guarantees a minimum ex-ante probability of γ for opening each box while trying to minimize the mana used. We show that for $\gamma \leq 1 - \frac{1}{\sqrt{k}}$ this algorithm never requires more than k units of mana.

Definition 4 (γ-Conservative Magician). *The magician adaptively computes a sequence of thresholds $\theta_1, \theta_2, \ldots \in \mathbb{R}_+$ and makes a decision about each box as follows: let W_i denote the amount of mana lost prior to seeing the i^{th} box; the magician makes a decision about box i by comparing W_i against θ_i; if $W_i < \theta_i$, it opens the box; if $W_i > \theta_i$, it does not open the box; and if $W_i = \theta_i$, it randomizes and opens the box with some probability (to be defined). The magician chooses the smallest threshold θ_i for which $\mathbf{Pr}[W_i \leq \theta_i] \geq \gamma$ where the probability is computed ex ante (i.e., not conditioned on $X_1, \ldots, X_{i-1}$). Note that γ is a parameter that is given. Let $F_{W_i}(w) = \mathbf{Pr}[W_i \leq w]$ denote the ex ante CDF of random variable W_i, and let Y_i be the indicator random variable which is 1 iff the magician opens the box i. Formally, the probability with which the magician should open box i conditioned on W_i is computed as follows*[12].

$$\mathbf{Pr}[Y_i = 1 | W_i] = \begin{cases} 1 & W_i < \theta_i \\ (\gamma - F^-_{W_i}(\theta_i))/(F_{W_i}(\theta_i) - F^-_{W_i}(\theta_i)) & W_i = \theta_i \\ 0 & W_i > \theta_i \end{cases} \qquad (Y)$$

$$\theta_i = \inf\{w | F_{W_i}(w) \geq \gamma\} \qquad (\theta)$$

In the above definition, $F^-_{W_i}$ is the left limit of F_{W_i}, i.e., $F^-_{W_i}(w) = \mathbf{Pr}[W_i < w]$.

Note that $F_{W_{i+1}}$ and $F^-_{W_{i+1}}$ are fully determined by F_{W_i} and F_{X_i} and the choice of γ (see Theorem 3). Observe that θ_i is in fact computed before seeing box i itself.

A γ-conservative magician may fail for a choice of γ unless all thresholds θ_i are less than or equal to $k-1$. The following theorem states a condition on γ that is sufficient to guarantee that $\theta_i \leq k-1$ for all i.

Theorem 2 (γ-Conservative Magician). *For any $\gamma \leq 1 - \frac{1}{\sqrt{k}}$, a γ-conservative magician with k units of mana opens each box with an ex ante probability of γ exactly.*

Proof. See Section 5.

Definition 5 (γ_k). *We define γ_k to be the largest probability such that for any $k' \geq k$ and any instance of the magician's problem with k' units of mana, the thresholds computed by a γ_k-conservative magician are no more than $k'-1$. In other words, γ_k is the optimal choice of γ which works for all instances with $k' \geq k$ units of mana. By Theorem 2, we know that γ_k must be*[13] *at least $1 - \frac{1}{\sqrt{k}}$.*

Observe that γ_k is a non-decreasing function in k and approaches 1 as $k \to \infty$. However $\gamma_1 = 0$ which is in contrast to the bound of $\frac{1}{2}$ obtained for $k = 1$ in [1] in which all X_i are Bernoulli random variables. It turns out that when X_i are arbitrary random variables in $[0,1]$, no algorithm exists for the magician that can guarantee a constant non-zero probability for opening each box.

[12] Assume $W_0 = 0$.

[13] Because for any $k' \geq k$ obviously $1 - \frac{1}{\sqrt{k}} \leq 1 - \frac{1}{\sqrt{k'}}$.

Proposition 1. *For the generalized magician's problem for $k = 1$, no algorithm for the magician (online or offline) can guarantee a constant non-zero probability for opening each box.*

Proof. Suppose there is an algorithm for the magician that is guaranteed to open each box with a probability of at least $\gamma \in (0, 1]$. We construct an instance in which the algorithm fails. Let $n = \lceil \frac{1}{\gamma} \rceil + 1$. Suppose all X_i are (independently) drawn from the distribution specified below.

$$X_i = \begin{cases} \frac{1}{2n} & \text{with prob. } 1 - \frac{1}{2n} \\ 1 & \text{with prob. } \frac{1}{2n} \end{cases}, \qquad \forall i \in [n]$$

As soon as the magician opens a box, the remaining mana will be less than 1, so he will not be able to open any other box, i.e., the magician can open only one box at every instance. Let Y_i denote the indicator random variable which is 1 iff the magician opens box i. Since $\sum_i Y_i \leq 1$, it must be $\sum_i \mathbf{E}[Y_i] \leq 1$. On the other hand, $\mathbf{E}[Y_i] \geq \gamma$ because the magician has guaranteed to open each box with a probability of at least γ. However $\sum_i \mathbf{E}[Y_i] \geq n\gamma > 1$ which is a contradiction. Note that $\sum_i \mathbf{E}[X_i] < 1$ so it satisfies the requirement of Definition 3.

Computation of $F_{W_i}(\cdot)$. For every $i \in [n]$, the equation $W_{i+1} = W_i + Y_i X_i$ relates the distribution of W_{i+1} to those of W_i and X_i [14]. The following lemma shows that the distribution of W_{i+1} is fully determined by the information available to the magician before seeing box $i + 1$.

Theorem 3. *In the algorithm of γ-conservative magician (Definition 4), the choice of γ and the distributions of $X_1, \ldots, X_i$ fully determine the distribution of W_{i+1}, for every $i \in [n]$. In particular, $F_{W_{i+1}}$ can be recursively defined as follows.*

$$F_{W_{i+1}}(w) = F_{W_i}(w) - G_i(w) + \mathbf{E}_{X_i \sim F_{X_i}}[G_i(w - X_i)] \quad \forall i \in [n], \forall w \in \mathbb{R}_+ \quad (F_W)$$
$$G_i(w) = \min(F_{W_i}(w), \gamma) \quad \forall i \in [n], \forall w \in \mathbb{R}_+ \quad (G)$$

Proof. See Section 5.

As a corollary of Theorem 3, we show how F_{W_i} can be computed using dynamic programming, assuming X_i can only take discrete values that are proper multiples of some minimum value.

Corollary 1. *If all X_i are proper multiple of $\frac{1}{D}$ for some $D \in \mathbb{N}$, then $F_{W_i}(\cdot)$ can be computed using the following dynamic program.*

$$F_{W_{i+1}}(w) = \begin{cases} F_{W_i}(w) - G_i(w) + \sum_\ell \mathbf{Pr}[X_i = \frac{\ell}{D}] G_i(w - \frac{\ell}{D}) & i \geq 1, w \geq 0 \\ 1 & i = 0, w \geq 0 \\ 0 & \textit{otherwise.} \end{cases}$$

for all values of $i \in [n]$ and $w \in \mathbb{R}_+$. In particular, the γ-conservative magician makes a decision for each box in time $O(D)$.

[14] Note that the distribution of Y_i is dependent on/determined by W_i.

Note that it is enough to compute F_{W_i} only for proper multiples of $\frac{1}{D}$ because $F_{W_i}(w) = F_{W_i}(\frac{\lfloor Dw \rfloor}{D})$ for any $w \in \mathbb{R}_+$.

4 The Online Algorithm

We present an online algorithm which obtains at least $1 - \frac{1}{\sqrt{k}}$-fraction of the optimal value of the linear program ($\overline{OPT}$). The algorithm uses, as a black box, the solution of the generalized magician's problem.

Definition 6 (Online Stochastic GAP Algorithm)

1. *Solve the linear program ($\overline{OPT}$) and let x be an optimal assignment.*
2. *For each $j \in [m]$, create a γ-conservative magician (Definition 4) with c_j units of mana for bin j. γ is a parameter that is given.*
3. *Upon arrival of each item $i \in [n]$, do the following:*
   (a) *Let t denote the type of item i.*
   (b) *Choose a bin at random such that each bin $j \in [m]$ is chosen with probability $\frac{x_{itj}}{p_{it}}$. Let j^* denote the chosen bin.*
   (c) *For each $j \in [m]$, define the random variable X_{ij} as $X_{ij} \leftarrow S_{itj}$ if $j^* = j$, and $X_{ij} \leftarrow 0$ otherwise[15]. For each $j \in [m]$, write the CDF of X_{ij} on a box and present it to the magician of bin j. The CDF of X_{ij} is $F_{X_{ij}}(s) = (1 - \sum_{t'} x_{it'j}) + \sum_{t'} x_{it'j} F_{it'j}(s)$.*
   (d) *If the magician for bin j^* opened his box in step 3c, then assign item i to bin j^*, otherwise discard the item. For each $j \in [m]$, if the magician of bin j opened his box in step 3c, decrease the mana of that magician by X_{ij}. In particular, $X_{ij} = 0$ for all $j \neq j^*$, and $X_{ij^*} = S_{itj^*}$.*

Theorem 4. *For any $\gamma \leq \gamma_k$, the online algorithm of Definition 6 obtains in expectation at least a γ-fraction of the expected value of the optimal offline assignment (recall that $\gamma_k \geq 1 - \frac{1}{\sqrt{k}}$).*

Proof. By Theorem 1, it is enough to show that the online algorithm obtains in expectation at least a γ-fraction of the optimal value of the linear program ($\overline{OPT}$). Let x be an optimal assignment for the LP. The contribution of each item $i \in [n]$ to the value of bin $j \in [m]$ in the LP is exactly $\sum_t v_{itj} x_{itj}$. We show that the online algorithm obtains in expectation $\gamma \sum_t v_{itj} x_{itj}$ from each item i and each bin j.

Consider an arbitrary item $i \in [n]$ and an arbitrary bin $j \in [m]$. WLOG, suppose the items are indexed in the order in which they arrive. Observe that

$$\mathbf{E}[X_{ij}] = \sum_t p_{it} \frac{x_{itj}}{p_{it}} \mathbf{E}[S_{itj}] = \sum_t x_{itj} \widetilde{s}_{itj}.$$

Consequently, $\sum_i \mathbf{E}[X_{ij}] = \sum_i \sum_t x_{itj} \widetilde{s}_{itj} \leq c_j$.

[15] Note that S_{itj} is learned only after item i is placed in bin j which implies that X_{ij} may not be known at this point, however the algorithm does not use X_{ij} until after it is learned.

The last inequality follows from the first set of constraints in the LP of ($\overline{OPT}$). Given that $\sum_i \mathbf{E}[X_{ij}] \leq c_j$ and $\gamma \leq \gamma_k \leq \gamma_{c_j}$, Theorem 2 implies that the magician of bin j opens each box with a probability of γ. Therefore, the expected contribution of item i to bin j is exactly $\sum_t \gamma p_{it} \frac{x_{itj}}{p_{it}} v_{itj} = \gamma \sum_t x_{itj} v_{itj}$. Consequently, the online algorithm obtains $\gamma \sum_i \sum_j \sum_t x_{itj} v_{itj}$ in expectation which is at least a γ-fraction of the expected value of the optimal offline assignment. Furthermore, each magician guarantees that the total size of the items assigned to each bin does not exceed the capacity of that bin.

5 Analysis of Generalized γ-Conservative Magician

We present the proof of Theorem 2. We prove the theorem in two parts. In the first part, we show that the thresholds computed by the γ-conservative magician indeed guarantee that each box is opened with an ex-ante probability of γ, assuming there is enough mana. In the second part, we show that for any $\gamma \leq 1 - \frac{1}{\sqrt{k}}$, the thresholds θ_i are less than or equal to $k-1$, for all i, which implies that the magician never requires more than k units of mana. Below, we repeat the formulation of the threshold based strategy of the magician.

$$\mathbf{Pr}\left[Y_i = 1 | W_i\right] = \begin{cases} 1 & W_i < \theta_i \\ (\gamma - F^-_{W_i}(\theta_i))/(F_{W_i}(\theta_i) - F^-_{W_i}(\theta_i)) & W_i = \theta_i \\ 0 & W_i > \theta_i \end{cases} \qquad (Y)$$

$$\theta_i = \inf\{w | F_{W_i}(w) \geq \gamma\} \qquad (\theta)$$

Part 1. We show that the thresholds computed by a γ-conservative magician guarantee that each box is opened with an ex ante probability of γ (i.e., $\mathbf{Pr}[Y_i = 1] = \gamma$), assuming there is enough mana.

$$\begin{aligned} \mathbf{Pr}\left[Y_i \leq w\right] &= \mathbf{Pr}\left[Y_i = 1 \cap W_i < \theta_i\right] + \mathbf{Pr}\left[Y_i = 1 \cap W_i = \theta_i\right] \\ &\quad + \mathbf{Pr}\left[Y_i = 1 \cap W_i > \theta_i\right] \\ &= \mathbf{Pr}\left[W_i < \theta_i\right] + \frac{\gamma - F^-_{W_i}(\theta_i)}{F_{W_i}(\theta_i) - F^-_{W_i}(\theta_i)} \mathbf{Pr}\left[W_i = \theta_i\right] = \gamma \end{aligned}$$

Part 2. Assuming $\gamma \leq 1 - \frac{1}{\sqrt{k}}$, we show that the thresholds computed by a γ-conservative magician are no more than $k-1$ (i.e., $\theta_i \leq k-1$ for all i). First, we present an interpretation of how $F_{W_i}(\cdot)$ evolves in i in terms of a sand displacement process.

Definition 7 (Sand Displacement Process). *Consider one unit of infinitely divisible sand which is initially at position* 0 *on the real line. The sand is gradually moved to the right and distributed over the real line in* n *rounds. Let* $F_{W_i}(w)$ *denote the total amount of sand in the interval* $[0, w]$ *at the beginning of round* $i \in [n]$*. At each round* i *the following happens.*

(I) The leftmost γ-fraction of the sand is selected by first identifying the smallest threshold $\theta_i \in \mathbb{R}_+$ such that $F_{W_i}(\theta_i) \geq \gamma$ and then selecting all the sand in the interval $[0, \theta_i)$ and selecting a fraction of the sand at position θ_i itself such that the total amount of selected sand is equal to γ. Formally, if $G_i(w)$ denotes the total amount of sand selected from $[0, w]$, the selection of sand is such that $G_i(w) = \min(F_{W_i}(w), \gamma)$, for every $w \in \mathbb{R}_+$. In particular, this implies that only a fraction of the sand at position θ_i itself might be selected, however all the sand to the left of position θ_i is selected.

(II) The selected sand is moved to the right as follows. Consider the given random variable $X_i \in [0, 1]$ and let $F_{X_i}(\cdot)$ denote its CDF. For every point $w \in [0, \theta_i]$ and every distance $\delta \in [0, 1]$, take a fraction proportional to $\mathbf{Pr}[X_i = \delta]$ out of the sand which was selected from position w and move it to position $w + \delta$.

It is easy to see that θ_i and $F_{W_i}(w)$ resulting from the above process are exactly the same as those computed by the γ-conservative magician.

Lemma 1. *At the end of the i^{th} round of the sand displacement process, the total amount of sand in the interval $[0, w]$ is given by the following equation.*

$$F_{W_{i+1}}(w) = F_{W_i}(w) - G_i(w) + \mathbf{E}_{X_i \sim F_{X_i}}\left[G_i(w - X_i)\right] \quad \forall i \in [n], \forall w \in \mathbb{R}_+ \quad (F_W)$$

Proof. According to definition of the sand displacement process, $F_{W_{i+1}}(w)$ can be defined as follows.

$$\begin{aligned} F_{W_{i+1}}(w) &= (F_{W_i}(w) - G_i(w)) + \iint_{\omega+\delta \leq w} dG_i(\omega)\, dF_{X_i}(\delta) \\ &= F_{W_i}(w) - G_i(w) + \int G_i(\omega - \delta)\, dF_{X_i}(\delta) \\ &= F_{W_i}(w) - G_i(w) + \mathbf{E}_{X_i \sim F_{X_i}}\left[G_i(w - X_i)\right] \end{aligned}$$

Proof (Proof of Theorem 3). The claim follows directly from Lemma 1

Consider a conceptual barrier which is at position $\theta_i + 1$ at the beginning of round i and is moved to position $\theta_{i+1} + 1$ for the next round, for each $i \in [n]$. It is easy to verify (i.e., by induction) that the sand never crosses to the right side of the barrier (i.e., $F_{W_{i+1}}(\theta_i + 1) = 1$). In what follows, the sand theorem implies that the sand remains concentrated near the barrier throughout the process. The barrier theorem implies that the barrier never passes k.

Theorem 5 (Sand). *Throughout the sand displacement process (Definition 7), at the beginning of round $i \in [n]$, the following inequality holds.*

$$F_{W_i}(w) < \gamma F_{W_i}(w + 1), \qquad \forall i \in [n], \forall w \in [0, \theta_i) \qquad (F_W\text{-ineq})$$

Furthermore, at the beginning of round $i \in [n]$, the average distance of the sand from the barrier, denoted by d_i, is upper bounded by the following inequalities[16] *in which the first inequality is strict except for $i = 1$.*

[16] Note that $\{z\} = z - \lfloor z \rfloor$, for any z.

$$d_i \le (1-\{\theta_i\})\frac{1-\gamma^{\lfloor\theta_i\rfloor+1}}{1-\gamma} + \{\theta_i\}\,\frac{1-\gamma^{\lceil\theta_i\rceil+1}}{1-\gamma} \le \frac{1-\gamma^{\lceil\theta_i\rceil+1}}{1-\gamma} < \frac{1}{1-\gamma}, \quad \forall i \in [n] \tag{d}$$

Proof. We start by proving the inequality (F_W-ineq). The proof is by induction on i. The case of $i = 1$ is trivial because all the sand is at position 0 and so $\theta_1 = 0$. Suppose the inequality holds at the beginning of round i for all $w \in [0, \theta_i)$; we show that it holds at the beginning of round $i+1$ for all $w \in [0, \theta_{i+1})$. Note that $\theta_i \le \theta_{i+1} \le \theta_i + 1$, so there are two possible cases:

Case 1. $w \in [0, \theta_i)$. Observe that $G_i(w) = F_{W_i}(w)$ in this interval, so:

$$\begin{aligned}
F_{W_{i+1}}(w) &= F_{W_i}(w) - G_i(w) + \mathbf{E}_{X_i}\left[G_i(w - X_i)\right] && \text{by } (F_W).\\
&= \mathbf{E}_{X_i}\left[F_{W_i}(w - X_i)\right] && \text{by } G_i(w) = F_{W_i}(w), \text{ for } w \in [0,\theta_i).\\
&< \mathbf{E}_{X_i}\left[\gamma F_{W_i}(w - X_i + 1)\right] && \text{by induction hypothesis.}\\
&= \gamma\,\mathbf{E}_{X_i}\left[F_{W_i}(w - X_i + 1) - G_i(w - X_i + 1) + G_i(w - X_i + 1)\right]\\
&\le \gamma\ \left(F_{W_i}(w+1) - G_i(w+1) + \mathbf{E}_{X_i}\left[G_i(w - X_i + 1)\right]\right)\\
&\qquad\qquad \text{by monotonicity of } F_{W_i}(\cdot) - G_i(\cdot).\\
&= \gamma F_{W_{i+1}}(w+1) && \text{by } (F_W).
\end{aligned}$$

Case 2. $w \in [\theta_i, \theta_{i+1}]$. We prove the claim by showing that $F_{W_{i+1}}(w) < \gamma$ and $F_{W_{i+1}}(w+1) = 1$. Observe that $F_{W_{i+1}}(w) < \gamma$ because $w < \theta_{i+1}$ and because of the definition of θ_{i+1} in (θ). Furthermore, observe that $F_{W_{i+1}}(w+1) \ge F_{W_{i+1}}(\theta_i + 1) = 1$ both before and after round i all the sand is still contained in the interval $[0, \theta_i + 1]$.

Next, we prove inequality (d) which upper bounds the average distance of the sand from the barrier at the beginning of round $i \in [n]$.

$$\begin{aligned}
d_i &= \int_0^{\theta_i+1} (\theta_i + 1 - w)\, dF_{W_i}(w)\\
&= \int_0^{\theta_i+1} F_{W_i}(w)\, dw && \text{by integration by part.}\\
&= \sum_{\ell=0}^{\lceil\theta_i\rceil} \int_{\theta_i-\ell}^{\theta_i+1-\ell} F_{W_i}(w)\, dw\\
&\le \sum_{\ell=0}^{\lfloor\theta_i\rfloor} \int_{\theta_i}^{\theta_i+1} \gamma^\ell F_{W_i}(w)\, dw \ + \int_{\lfloor\theta_i\rfloor+1}^{\theta_i+1} \gamma^{\lceil\theta_i\rceil} F_{W_i}(w)\, dw && \text{by } (F_W\text{-ineq}).\\
&\le \sum_{\ell=0}^{\lfloor\theta_i\rfloor} \gamma^\ell \ + \{\theta_i\}\,\gamma^{\lceil\theta_i\rceil} && \text{by } F_{W_i}(w) \le 1.\\
&= (1-\{\theta_i\}) \sum_{\ell=0}^{\lfloor\theta_i\rfloor} \gamma^\ell \ + \{\theta_i\} \sum_{\ell=0}^{\lceil\theta_i\rceil} \gamma^\ell\\
&= (1-\{\theta_i\})\frac{1-\gamma^{\lfloor\theta_i\rfloor+1}}{1-\gamma} + \{\theta_i\}\,\frac{1-\gamma^{\lceil\theta_i\rceil+1}}{1-\gamma} \le \frac{1-\gamma^{\lceil\theta_i\rceil+1}}{1-\gamma}
\end{aligned}$$

The last inequality follows because $(1-\beta)L + \beta H \leq H$ for any $\beta \in [0,1]$ and any L, H with $L \leq H$. Note that at least one of the first two inequalities is strict except for $i = 1$ which proves the claim.

Theorem 6 (Barrier). *If $\sum_{i=1}^{n} \mathbf{E}_{X_i \sim F_{X_i}}[X_i] \leq k$ for some $k \in \mathbb{N}$, and $\gamma \leq 1 - \frac{1}{\sqrt{k}}$, then the distance of the barrier from the origin is no more than k throughout the process, i.e., $\theta_i \leq k-1$ for all $i \in [n]$.*

Proof. At the beginning of round i, let d_i and d'_i denote the average distance of the sand from the barrier and from the origin respectively. Recall that the barrier is defined to be at position $\theta_i + 1$ at the beginning of round i. Observe that $d_i + d'_i = \theta_i + 1$. Furthermore, $d'_{i+1} = d'_i + \gamma\, \mathbf{E}[X_i]$, i.e., the average distance of the sand from the origin is increased exactly by $\gamma\, \mathbf{E}[X_i]$ during round i (because the amount of selected sand is exactly γ and the sand selected from every position $w \in [0, \theta_i]$ is moved to the right by an expected distance of $\mathbf{E}[X_i]$). By applying Theorem 5 we get the following inequality for all $i \in [n]$.

$$\begin{aligned}\theta_i + 1 = d'_i + d_i &< \gamma \sum_{r=1}^{i-1} \mathbf{E}\left[X_i\right] + d_i \\ &\leq \gamma k + (1 - \{\theta_i\}) \frac{1 - \gamma^{\lfloor \theta_i \rfloor + 1}}{1-\gamma} + \{\theta_i\} \frac{1 - \gamma^{\lceil \theta_i \rceil + 1}}{1-\gamma}\end{aligned}$$

In order to show that distance of the barrier from the origin is no more than k throughout the process, it is enough to show that the above inequality cannot hold for $\theta_i > k-1$. In fact it is just enough to show that it cannot hold for $\theta_i = k-1$; alternatively, it is enough to show that the complement of the above inequality holds for $\theta_i = k-1$, i.e., $k \geq \gamma k + \frac{1-\gamma^k}{1-\gamma}$. To complete the proof, consider the the stronger inequality $k \geq \gamma k + \frac{1}{1-\gamma}$ which is quadratic in γ and can be solved to get a bound of $\gamma \leq 1 - \frac{1}{\sqrt{k}}$.

Theorem 6 implies that a γ-conservative magician requires no more than k units of mana, assuming that $\gamma \leq 1 - \frac{1}{\sqrt{k}}$. That completes the proof of Theorem 2.

References

1. Alaei, S.: Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers. In: FOCS (2011)
2. Alaei, S., Hajiaghayi, M.T., Liaghat, V.: Online prophet-inequality matching with applications to ad allocation. In: EC (2012)
3. Alaei, S., Hajiaghayi, M.T., Liaghat, V., Pei, D., Saha, B.: AdCell: Ad allocation in cellular networks. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 311–322. Springer, Heidelberg (2011)
4. Bhalgat, A.: A $(2 + \epsilon)$-approximation algorithm for the stochastic knapsack problem (2012) (unpublished manuscript)
5. Bhalgat, A., Goel, A., Khanna, S.: Improved approximation results for stochastic knapsack problems. In: SODA (2011)
6. Chekuri, C., Khanna, S.: A ptas for the multiple knapsack problem. In: SODA (2000)

7. Dean, B.C., Goemans, M.X., Vondrak, J.: Approximating the stochastic knapsack problem: The benefit of adaptivity. In: FOCS (2004)
8. Devanur, N.R., Jain, K., Sivan, B., Wilkens, C.A.: Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In: EC (2011)
9. Feige, U., Immorlica, N., Mirrokni, V., Nazerzadeh, H.: A combinatorial allocation mechanism with penalties for banner advertising. In: WWW (2008)
10. Feige, U., Vondrak, J.: Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. In: FOCS (2006)
11. Feldman, J., Korula, N., Mirrokni, V., Muthukrishnan, S., Pál, M.: Online ad assignment with free disposal. In: Leonardi, S. (ed.) WINE 2009. LNCS, vol. 5929, pp. 374–385. Springer, Heidelberg (2009)
12. Fleischer, L., Goemans, M.X., Mirrokni, V.S., Sviridenko, M.: Tight approximation algorithms for maximum general assignment problems. In: SODA (2006)
13. Goel, A., Indyk, P.: Stochastic load balancing and related problems. In: FOCS (1999)
14. Hajiaghayi, M.T., Kleinberg, R.D., Sandholm, T.: Automated online mechanism design and prophet inequalities. In: AAAI (2007)
15. Jain, K., Mahdian, M., Markakis, E., Saberi, A., Vazirani, V.V.: Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. J. ACM (2003)
16. Kleinberg, J., Rabani, Y., Tardos, E.: Allocating bandwidth for bursty connections. In: STOC (1997)
17. Krengel, U., Sucheston, L.: Semiamarts and finite values. Bull. Am. Math. Soc. (1977)
18. Krengel, U., Sucheston, L.: On semiamarts, amarts, and processes with finite value. In: Kuelbs, J. (ed.) Probability on Banach Spaces (1978)
19. Mehta, A., Saberi, A., Vazirani, U., Vazirani, V.: Adwords and generalized online matching. J. ACM (2007)
20. Wikipedia (2012), `http://en.wikipedia.org/wiki/mana`

On the NP-Hardness of Approximating Ordering Constraint Satisfaction Problems

Per Austrin, Rajsekar Manokaran, and Cenny Wenner

School of Computer Science and Communication,
KTH – Royal Institute of Technology, Stockholm, Sweden
{austrin,rajsekar,cenny}@csc.kth.se

Abstract. We show improved NP-hardness of approximating *Ordering Constraint Satisfaction Problems* (OCSPs). For the two most well-studied OCSPs, Maximum Acyclic Subgraph and Maximum Betweenness, we prove inapproximability of $14/15 + \varepsilon$ and $1/2 + \varepsilon$.

An OCSP is said to be approximation resistant if it is hard to approximate better than taking a uniformly random ordering. We prove that the Maximum *Non*-Betweenness Problem is approximation resistant and that there are width-m approximation-resistant OCSPs accepting only a fraction $1/(m/2)!$ of assignments. These results provide the first examples of approximation-resistant OCSPs subject only to $\mathsf{P} \neq \mathsf{NP}$.

Our reductions from Label Cover differ from previous works in two ways. First, we establish a somewhat general bucketing lemma permitting us to reduce the analysis of ordering predicates to that of classical predicates. Second, instead of "folding", which is not available for ordering predicates, we employ permuted instantiations of the predicates to limit the value of poorly correlated strategies.

1 Introduction

We study the NP-hardness of approximating a rich class of optimization problems known as the *Ordering Constraint Satisfaction Problems* (OCSPs). An instance of an OCSP is described by a set of variables $\mathcal{X}$ and a set of *local ordering constraints* $\mathcal{C}$. Each constraint specifies a set of variables and a set of permitted permutations of these variables. The objective is to find a permutation of $\mathcal{X}$ that maximizes the fraction of constraints satisfied by the induced local permutations.

A simple example of an OCSP is the Maximum Acyclic Subgraph (MAS) where one is given a directed graph $G = (V, A)$ with the task of finding an acyclic subgraph of G with the maximum number of edges. Phrased as an OCSP, V is the set of variables and each edge $u \to v$ is a constraint "$u \prec v$" dictating that u should precede v. The maximum fraction of constraints simultaneously satisfiable by an ordering of V is then exactly the normalized size of the largest acyclic subgraph, also called the value of the instance. Since the constraints in an MAS instance are on two variables, it is an OCSP of width 2. Another example of an OCSP is the Maximum Betweenness (Max

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 26–41, 2013.

BTW) problem; [11] a width-3 OCSP where a constraint on a triplet of variables (x, y, z) permits either $x \prec z \prec y$ or its reverse $y \prec z \prec x$; in other words, z has to be between x and y and hence the name for the problem.

Determining the value of a MAS instance is already NP-hard and one turns to approximation algorithms. A c-approximation algorithm for some $c \leq 1$ is one that outputs an ordering satisfying at least a $c \cdot \mathsf{val}(\mathcal{I})$ fraction of the constraints. Every OCSP admits a naive approximation algorithm that picks an ordering of $\mathcal{X}$ uniformly at random without even looking at the instance. For MAS, this algorithm yields a 1/2-approximation in expectation as each constraint is satisfied with probability 1/2. Surprisingly, there is evidence that this mindless procedure achieves the best approximation ratio possible in polynomial time: assuming Khot's Unique Games Conjecture (UGC) [16], MAS is hard to approximate within $1/2 + \varepsilon$ for every $\varepsilon > 0$ [13,12]. An OCSP is called *approximation resistant* if it exhibits this behavior, i.e., if it is NP-hard to improve on the guarantee of the random-ordering algorithm by ε for every $\varepsilon > 0$. In fact, the results of [12] are much more general: assuming the UGC, they prove that *every* OCSP of bounded width.

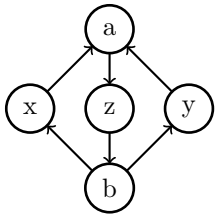


Fig. 1. An MAS instance with value 5/6

In many cases, such as for Vertex Cover, Max Cut, and as we just mentioned, for all OCSPs, assuming the UGC allows us to prove optimal inapproximability results, much stronger than known plain NP-hardness. For instance, the problems MAS and Max BTW were to date only known to be NP-hard to approximate within $65/66 + \varepsilon$ [18] and $47/48 + \varepsilon$ [9], which comes far from matching the random assignment thresholds of 1/2 and 1/3, respectively. In fact, while the UGC implies that all OCSPs are approximation resistant, there were no results proving NP-hard approximation resistance of an OCSP prior to this work. In contrast, there is a significant body of work on NP-hard approximation resistance of classical Constraint Satisfaction Problems (CSPs) [15,20,10,6]. Furthermore, the UGC is still very much open and recent algorithmic advances have given rise to subexponential algorithms for Unique Games [1,5] putting the conjecture in question. Several recent works have also been aimed at bypassing the UGC for natural problems by providing comparable results without assuming the conjecture [14,6].

1.1 Results

In this work we obtain improved NP-hardness of approximating various OCSPs. While a complete characterization such as in the UG regime still eludes us, our results improve the knowledge of what we believe are four important flavors of OCSPs; see Table 1 for a summary of the present state of affairs.

We address the two most studied OCSPs: MAS and Max BTW. For MAS, we show a factor $(14/15 + \varepsilon)$-inapproximability, improving the factor from

Table 1. Known results and our improvements

Problem	Approx. factor	UG-inapprox.	NP-inapprox.	This work
MAS	$1/2 + \Omega(1/\log n)$ [8]	$1/2 + \varepsilon$ [13]	$65/66 + \varepsilon$ [18]	$14/15 + \varepsilon$
Max BTW	$1/3$	$1/3 + \varepsilon$ [7]	$47/48 + \varepsilon$ [9]	$1/2 + \varepsilon$
Max NBTW	$2/3$	$2/3 + \varepsilon$ [7]	-	$2/3 + \varepsilon$
m-OCSP	$1/m!$	$1/m! + \varepsilon$ [12]	-	$1/\lfloor m/2 \rfloor! + \varepsilon$

$65/66 + \varepsilon$ [18]. For Max BTW, we show a factor $(1/2 + \varepsilon)$-inapproximability improving from $47/48 + \varepsilon$ [9].

Theorem 1. *For every $\varepsilon > 0$, it is NP-hard to distinguish between* MAS *instances with value at least $15/18 - \varepsilon$ from instances with value at most $14/18 + \varepsilon$.*

Theorem 2. *For every $\varepsilon > 0$, it is NP-hard to distinguish between* Max BTW *instances with value at least $1 - \varepsilon$ from instances with value at most $1/2 + \varepsilon$.*

The above two results are inferior to what is known assuming the UGC and in particular do not prove approximation resistance. We introduce the Maximum Non-Betweenness (Max NBTW) problem which accepts the *complement of the predicate* in Max BTW. This predicate accepts 4 of the 6 permutations on three elements and thus a random ordering satisfies $2/3$ of the constraints in expectation. We show that this is optimal up to smaller-order terms.

Theorem 3. *For every $\varepsilon > 0$, it is NP-hard to distinguish between* Max NBTW *instances with value at least $1 - \varepsilon$ from instances with value at most $2/3 + \varepsilon$.*

Finally, we address the approximability of a generic width-m OCSP as a function of the width m. In the CSP world, the generic version is called m-CSP and we call the ordering version m-OCSP. We devise a simple predicate, "$2t$-Same Order" ($2t$-SO) on $m = 2t$ variables that is satisfied only if the first t elements are relatively ordered exactly as the last t elements. A random ordering satisfies only a fraction $1/t!$ of the constraints and we prove that this is essentially optimal, implying a $(1/\lfloor m/2 \rfloor! + \varepsilon)$-factor inapproximability of m-OCSP.

Theorem 4. *For every $\varepsilon > 0$ and integer $m \geq 2$, it is NP-hard to distinguish m-OCSP instances with value at least $1 - \varepsilon$ from value at most $1/\lfloor m/2 \rfloor! + \varepsilon$.*

1.2 Proof Overview

With the exception of MAS, our results follow a route which is by now standard in inapproximability: starting from the optimization problem Label Cover (LC), we give a reduction to the problem at hand using a *dictatorship-test* gadget, also known as a long-code test. We describe this reduction in the context of Max NBTW to highlight the new techniques in this paper.

The reduction produces an instance $\mathcal{I}$ of Max NBTW from an instance $\mathcal{L}$ of Label Cover (LC) such that $\mathsf{val}(\mathcal{I}) > 1-\eta$ if $\mathsf{val}(\mathcal{L}) = 1$ while $\mathsf{val}(\mathcal{I}) < 2/3+\eta$ if $\mathsf{val}(\mathcal{L}) \leq \delta$. By the PCP Theorem and the Parallel Repetition Theorem [3,2,19], it

is NP-hard to distinguish between $\mathsf{val}(\mathcal{L}) = 1$ and $\mathsf{val}(\mathcal{L}) \leq \delta$ for every constant $\delta > 0$ and thus we obtain the result in Theorem 3. The core component in this paradigm is the design of a dictatorship test: a Max NBTW instance on $[q]^L \cup [q]^R$, for integers q and label sets L and R. Let π be a map $R \to L$. Each constraint is a tuple $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ where $\mathbf{x} \in [q]^L$, while $\mathbf{y}, \mathbf{z} \in [q]^R$. The distribution of tuples is obtained as follows. First, pick $\mathbf{x}$, and $\mathbf{y}$ uniformly at random from $[q]^L$, and $[q]^R$. Set $z_j = y_j + x_{\pi(j)} \bmod q$. Finally, add noise by independently replacing each coordinate x_i, y_j and z_j with a uniformly random element from $[q]$ with probability γ.

This test instance has canonical assignments that satisfy almost all the constraints. These are obtained by picking an arbitrary $j \in [R]$, and partitioning the variables into q sets $S_0, \ldots S_{q-1}$ where $S_t = \{\mathbf{x} \in [q]^L |\ x_{\pi(j)} = t\} \cup \{\mathbf{y} \in [q]^R |\ y_j = t\}$. If a constraint $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is so that $\mathbf{x} \in S_t$, $\mathbf{y} \in S_u$ then $\mathbf{z} \notin S_v$ for any $v \in \{t+1, \ldots, u-1\}$ except with probability $O(\gamma)$. This is because, $(a+b) \bmod q$ is never strictly between a and b. Further, the probability that any two of $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ fall in the same set S_i is simply the probability that any two of $x_{\pi(j)}, y_j, z_j$ are assigned the same value, which is at most $O(1/q)$. Thus, ordering the variables so that $S_0 \prec S_1 \prec \ldots \prec S_{q-1}$ with an arbitrary ordering of the variables within a set satisfies a fraction $1 - O(1/q) - O(\gamma)$ constraints.

The proof of Theorem 3 requires a partial converse of the above: every ordering that satisfies more than a fraction $2/3 + \varepsilon$ of the constraints is more-or-less an ordering that depends on a few coordinates j as above. This proof involves three steps. First, we show that there is a $\Gamma = \Gamma(q, \gamma, \beta)$ such that every ordering $\mathcal{O}$ of $[q]^L$ or $[q]^R$ can be broken into Γ sets $S_0, \ldots, S_{\Gamma-1}$ such that one achieves expected value at least $\mathsf{val}(\mathcal{O}) - \beta$ for arbitrarily small β by ordering the sets $S_0 \prec \ldots \prec S_{\Gamma-1}$ and within each set ordering elements randomly. The proof of this "bucketing" uses hypercontractivity of noised functions from a finite domain. We note that a related bucketing argument is used in proving inapproximability of OCSPs assuming the UGC [13,12]. Their bucketing argument is tied to the use of the UGC, where $|L| = |R|$ for the corresponding dictatorship test, and does not extend to our setting. In particular, our approach yields a $\Gamma \gg q$ while they crucially require a $\Gamma \ll q$ in their work. We believe that our bucketing argument is more general and a useful primitive.

Then, similarly to [13,12], the bucketing argument allows an OCSP to be analyzed as if it were a CSP on a finite domain, enabling us to use powerful techniques for this common setting. In particular, one can argue that unless $\mathsf{val}(\mathcal{L}) > \delta$, the distribution of constraints $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ can be regarded as obtained by sampling $\mathbf{x}$ independently; in other words, $\mathbf{x}$ is "decoupled" from $(\mathbf{y}, \mathbf{z})$. We note that the marginal distribution of the tuple $(\mathbf{y}, \mathbf{z})$ is already symmetric with respect to swaps: $\mathbf{P}(\mathbf{y} = y, \mathbf{z} = z) = \mathbf{P}(\mathbf{y} = z, \mathbf{z} = y)$. In order to prove approximation resistance, we combine three of these dictatorship tests: the jth variant has $\mathbf{x}$ as the jth component of the 3-tuple. We show that the combined instance is symmetric with respect to every swap up to an η error unless $\mathsf{val}(\mathcal{L}) > \delta$. This implies that the instance has value at most $2/3 + O(\eta)$ hence proving approximation resistance of Max NBTW.

For Max BTW and Max $2t$-SO, we do not require the final symmetrization and instead use a dictatorship test based on a different distribution. Finally, the reduction to MAS is a simple gadget reduction from Max NBTW. For hardness results of width-two predicates, such gadget reductions presently dominate the scene of classical CSPs and also define the state of affairs for MAS. As an example, the best-known NP-hard approximation hardness of $16/17+\varepsilon$ for Max Cut is via a gadget reduction from Max 3-Lin-2 [15,21]. The previously best approximation hardness of MAS was also via a gadget reduction from Max 3-Lin-2 [18], although with the significantly smaller gap $65/66+\varepsilon$. By reducing from a problem more similar to MAS, namely Max NBTW, we improve to the approximation hardness to $14/15+\varepsilon$. The gadget in question is quite simple and we have in fact already seen it in Sect. 1.

Organization. Section 2 sets up the notation used in the rest of the article. Section 3 gives a general hardness result based on a test distribution which is subsequently used in Sect. 4 to derive our main results. The proof of the soundness of the general hardness reduction is largely given in Sect. 5.

2 Preliminaries

We denote by $[n]$ the integer interval $\{0,\dots,n-1\}$. Given a tuple of reals $\boldsymbol{x} \in \mathbb{R}^m$, we write $\boldsymbol{\sigma}(\boldsymbol{x}) \in S_m$ for the natural-order permutation on $\{1,\dots,m\}$ induced by $\boldsymbol{x}$. For a distribution $\mathcal{D}$ over $\Omega_1 \times \cdots \times \Omega_m$, we use $\mathcal{D}_{\leq t}$ and $\mathcal{D}_{>t}$ to denote the projection to coordinates up to t and the rest, respectively.

Ordering Constraint Satisfaction Problems. We are concerned with predicates $\mathcal{P} : S_m \to [0,1]$ on the symmetric group S_m. Such a predicate specifies a width-m OCSP written as $\mathrm{OCSP}(\mathcal{P})$. An instance $\mathcal{I}$ of $\mathrm{OCSP}(\mathcal{P})$ problem is a tuple $(\mathcal{X},\mathcal{C})$ where $\mathcal{X}$ is the set of *variables* and $\mathcal{C}$ is a distribution over ordered m-tuples of $\mathcal{X}$ referred to as the *constraints*.

An assignment to $\mathcal{I}$ is an injective map $\mathcal{O} : \mathcal{X} \to \mathbb{Z}$ called an ordering of $\mathcal{X}$. For a tuple $\boldsymbol{c} = (v_1,\dots,v_m)$, $\mathcal{O}_{|c}$ denotes the tuple $(\mathcal{O}(v_1),\dots,\mathcal{O}(v_m))$. An ordering is said to satisfy the constraint c when $\mathcal{P}(\boldsymbol{\sigma}(\mathcal{O}_{|c})) = 1$. The value of an ordering is the probability that a random constraint $c \leftarrow \mathcal{C}$ is satisfied by $\mathcal{O}$ and the value $\mathsf{val}(\mathcal{I})$ of an instance is the maximum value of any ordering. Thus,

$$\mathsf{val}(\mathcal{I}) = \max_{\mathcal{O}:\mathcal{X}\to\mathbb{Z}} \mathsf{val}(\mathcal{O};\mathcal{I}) = \max_{\mathcal{O}:\mathcal{X}\to\mathbb{Z}} \mathop{\mathbf{E}}_{c\in\mathcal{C}} \left[\mathcal{P}(\mathcal{O}_{|c})\right].$$

We extend the definition of value to include orderings that are not strictly injective as follows. Extend the predicate to $\mathcal{P} : \mathbb{Z}^m \to [0,1]$ by setting $\mathcal{P}(a_1,\dots,a_m) = \mathbf{E}_{\boldsymbol{\sigma}}[\mathcal{P}(\boldsymbol{\sigma})]$ where $\boldsymbol{\sigma}$ is drawn uniformly at random over all permutations in S_m such that $\sigma_i < \sigma_j$ whenever $a_i < a_j$. Note that the value of an instance is unchanged by this extension as there is always a complete ordering that attains the value of a non-injective map.

We define the predicates and problems studied in this work. MAS is exactly $\mathrm{OCSP}(\{(1,2)\})$. The betweenness predicate BTW is the set $\{(1,3,2),(3,1,2)\}$

and NBTW is $S_3 \setminus$ BTW. We define Max BTW as OCSP(BTW) and Max NBTW as OCSP(NBTW). Define $2t$-SO as the subset of S_{2t} such that the induced ordering on the first t elements equals that on the last t elements, i.e.

$$2t\text{-SO} \stackrel{\text{def}}{=} \{\pi \in S_{2t} \mid \sigma(\pi(1), \ldots, \pi(r)) = \sigma(\pi(r+1), \ldots, \pi(2t))\}.$$

This predicate has $^{(2t)!}/_{t!}$ elements and will be used in proving the inapproximability of the generic m-OCSP with constraints of width at most m.

Label Cover and Inapproximability. The problem LC is a common starting point of strong inapproximability results. An LC instance $\mathcal{L} = (U, V, E, L, R, \Pi)$ consists of a bi-partite graph $(U \cup V, E)$ associating with every edge u, v a projection $\pi_{uv} : R \to L$ with the goal of labeling the vertices $\lambda : U \cup V \to L \cup R$ to maximize the fraction of projections s.t. "$\pi_{uv}(\lambda(v)) = \lambda(u)$". The following theorem follows from the PCP Theorem [2] and parallel repetition [19].

Theorem 5. *For every $\varepsilon > 0$, there exists fixed label sets L and R such that it is* NP*-hard to distinguish LC instances of value 1 from instances of value at most ε.*

2.1 Primer on Real Analysis

We state facts about functions over a finite domain Ω taking values in the reals, that will be necessary in our analysis. We refer to a finite domain Ω along with a distribution μ as a probability space. Given a probability space (Ω, μ), the n^{th} tensor power is $(\Omega^n, \mu^{\otimes n})$ where $\mu^{\otimes n}(\omega_1, \ldots, \omega_n) = \mu(\omega_1) \cdots \mu(\omega_n)$. The ℓ_p norm of $f : \Omega \to \mathbb{R}$ w.r.t. μ is denoted by $\|f\|_{\mu,p}$ and is defined as $\mathbf{E}_{\mathbf{x}\sim\mu}\left[|f(\mathbf{x})|^p\right]^{1/p}$ for real $p \geq 1$ and $\max_{x \in \text{supp}(\mu)} f(x)$ for $p = \infty$. When clear from the context, we shall omit the distribution μ. The so-called noise operator and its properties play a pivotal role in our analysis.

Definition 1. *Let (Ω, μ) be a probability space and $f : \Omega^n \to \mathbb{R}$ be a function on the n^{th} tensor power. For a parameter $\rho \in [0, 1]$, the* noise operator T_ρ *takes f to $\mathrm{T}_\rho f \to \mathbb{R}$ defined by*

$$\mathrm{T}_\rho f(\boldsymbol{x}) = \mathbf{E}\left[f(\mathbf{y}) | \boldsymbol{x}\right],$$

where the ith coordinate of $\mathbf{y}$ is chosen as $\mathbf{y}_i = \boldsymbol{x}_i$ with probability ρ and otherwise as an independent new sample.

The noise operator preserves the mass $\mathbf{E}[f]$ of a function while spreading it in the space. The quantitative bound on this is referred to as hypercontractivity.

Theorem 6 ([23]; Theorem 3.16**,** 3.17 **of [17]).** *Let (Ω, μ) be a probability space in which the minimum nonzero probability of any atom is $\alpha < 1/2$. Then, for every $q > 2$ and every $f : \Omega^n \to \mathbb{R}$,*

$$\left\|\mathrm{T}_{\rho(q)} f\right\|_q \leq \|f\|_2,$$

where for $\alpha < 1/2$ *we set* $A = \frac{1-\alpha}{\alpha}$*;* $1/q' = 1 - 1/q$*; and* $\rho(q,\alpha) = \left(\frac{A^{1/q}-A^{-1/q}}{A^{1/q'}-A^{-1/q'}}\right)^{1/2}$*. For* $\alpha = 1/2$*, we set* $\rho(q) = (q-1)^{-1/2}$*.*

For a fixed probability space, the above theorem says that for every $\gamma > 0$, there is a $q > 2$ such that $\|\mathrm{T}_{1-\gamma} f\|_q \leq \|f\|_2$. For our application, we need the easy corollary that the reverse direction also holds: for every $\gamma > 0$, there exists a $q > 2$ such that hypercontractivity to the ℓ_2-norm holds.

Lemma 1. *Let* (Ω, μ) *be a probability space in which the minimum nonzero probability of any atom is* $\alpha \leq 1/2$*. Then, for every* $f : \Omega^n \to \mathbb{R}$*, small enough* $\gamma > 0$*,*

$$\|\mathrm{T}_{1-\gamma} f\|_{2+\delta} \leq \|f\|_2$$

for any $0 < \delta \leq \delta(\gamma,\alpha) = 2^{\frac{\log((1-\gamma)^{-2})-1}{\log(A)}}$ *with* $A = \frac{1-\alpha}{\alpha} > 1$*. Further,* $\delta(\gamma, 1/2) = \gamma(2-\gamma)(1-\gamma)^{-2}$*.*

Proof. The estimate for $\delta(\gamma, 1/2)$ follows immediately from the above theorem assuming $\gamma < 1/2$. In the case when $\alpha < 1/2$, solving $\rho^2 \stackrel{\text{def}}{=} (1-\gamma)^2 = (A^{1/q} - A^{-1/q})(A^{1-1/q} - A^{1/q-1})^{-1}$ for q gives, for $\gamma < 1 - A^{-1/2}$,

$$\delta = q - 2 = \frac{2\log(A)}{\log(\frac{1+\rho^2 A}{1+\rho^2/A})} - 2 \geq 2\frac{\log((1-\gamma)^{-2}) - 1}{\log(A)}. \qquad \square$$

3 A General Hardness Result

In this section, we prove a general inapproximability for OCSPs that, similar to results for classic CSPs, permit us to deduce hardness of approximation based on the existence of certain simple distributions. The proof is via a scheme of reductions from LC to OCSPs. For an m-width predicate $\mathcal{P}$, we instantiate this scheme with a distribution $\mathcal{D}$ over $Q_1^t \times Q_2^{m-t}$, for some parameters t, Q_1, and Q_2 to obtain a reduction to OCSP($\mathcal{P}$) instances. Theorems 2 to 4 follow from straightforward applications of this result using appropriate distributions.

The reduction itself is composed of pieces known as dictatorship test which is described in the next section. Section 3.2 uses this test to construct the overall reduction and also contains the properties of this reduction.Throughout this section, we assume $\mathcal{P}$ is the m-width predicate of interest and that $\mathcal{D}$ is the distribution of the appropriate signature.

3.1 Dictatorship Test

The dictatorship test uses a distribution parametrized by γ, and π and is denoted by $\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})$; its definition follows.

Procedure 1 (Test Distribution).

Parameters:

- *distribution* $\mathcal{D}$ *over* $\overbrace{Q_1 \times \ldots \times Q_1}^{t} \times \overbrace{Q_2 \times \ldots \times Q_2}^{m-t}$*;*
- *noise probability,* $\gamma > 0$*;*
- *projection map* $\pi : R \to L$*;*

Output: *Distribution* $\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})$ *over*

$$(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t)}, \mathbf{y}^{(t+1)}, \ldots, \mathbf{y}^{(m)}) \in \left(Q_1^L \times \cdots \times Q_1^L\right) \times \left(Q_2^R \times \cdots \times Q_2^R\right).$$

1. *pick a random* $|L| \times t$ *matrix* $\mathbf{X}$ *over* Q_1 *by letting each row be a sample from* $\mathcal{D}_{\leq t}$*, independently.*
2. *pick a random* $|R| \times (m-t)$ *matrix* $\mathbf{Y} \stackrel{\text{def}}{=} (\mathbf{y}^{(t+1)}, \ldots, \mathbf{y}^{(m)})$ *over* Q_2 *by letting the* i*:th row be a sample from* $\mathcal{D}_{>t}$ *conditioned on* $\mathcal{D}_{\leq t} = \mathbf{X}_{\pi(i)} = (\mathbf{x}^{(1)}_{\pi(i)}, \ldots, \mathbf{x}^{(t)}_{\pi(i)})$*.*
3. *for each entry of* $\mathbf{X}$ *(resp.* $\mathbf{Y}$*) independently, replace it with a sample from* Q_1 *(resp.* Q_2*) with probability* γ*.*
4. *output* $(\mathbf{X}, \mathbf{Y})$*.*

Recall our convention from Sect. 2 of extending $\mathcal{P} : \mathbb{Z}^m \to [0,1]$. For a pair of functions (f, g), we denote the tuple $(f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(t)}), g(\mathbf{y}^{(t+1)}), \ldots, g(\mathbf{y}^{(m)}))$ by $(f, g) \circ (\mathbf{X}, \mathbf{Y})$. Then, the *acceptance probability* on $\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})$ for a pair of functions (f, g) where $f : Q_1^L \to \mathbb{Z}$ and $g : Q_2^R \to \mathbb{Z}$ is:

$$\mathsf{Acc}_{f,g}(\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{X},\mathbf{Y}) \leftarrow \mathcal{T}_\pi^{(\gamma)}(\mathcal{D})} [\mathcal{P}((f, g) \circ (\mathbf{X}, \mathbf{Y}))]. \tag{1}$$

This definition is motivated by the overall reduction described in the next section. The distribution is designed so that functions (f, g) that are dictated by a single coordinate have a high acceptance probability, justifying the name of the test.

Lemma 2. *Let* $g : Q_2^R \to \mathbb{Z}$ *and* $f : Q_1^L \to \mathbb{Z}$ *be defined by* $g(\boldsymbol{y}) = y_i$ *and* $f(\mathbf{x}) = x_{\pi(i)}$ *for some* $i \in R$*. Then,* $\mathsf{Acc}_{f,g}(\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})) \geq \mathbf{E}_{\mathbf{x} \sim \mathcal{D}}[\mathcal{P}(\mathbf{x})] - \gamma m$*.*

Proof. The vector $(f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(t)}), g(\mathbf{y}^{(t+1)}), \ldots, g(\mathbf{y}^{(m)}))$ simply equals the $\pi(i)$:th row of $\mathbf{X}$ followed by the i:th row of $\mathbf{Y}$. With probability $(1-\gamma)^m \geq 1-\gamma m$ this is a sample from $\mathcal{D}$ and is hence accepted by $\mathcal{P}$ with probability at least $\mathbf{E}_{\mathbf{x} \sim \mathcal{D}}[\mathcal{P}(\mathbf{x})] - \gamma m$. □

We prove a partial converse of the above: unless f and g have influential coordinates i and j such that $\pi(j) = i$, the distribution $\mathcal{D}$ can be replaced by a product of two distributions with a negligible loss in the acceptance probability. We define this product distribution below and postpone the analysis to Sect. 5.2 in order to complete the description of the reduction.

Definition 2. *Given the base distribution* $\mathcal{D}$*, the* decoupled *distribution* $\mathcal{D}^\perp$ *is obtained by taking independent samples* $\mathbf{x}$ *from* $\mathcal{D}_{\leq t}$ *and* $\mathbf{y}$ *from* $\mathcal{D}_{>t}$*.*

3.2 Reduction from Label Cover

Procedure 2 (Reduction $R^{(\mathcal{P})}_{\mathcal{D},\gamma}$).
Parameters: *distribution $\mathcal{D}$ over $Q_1^t \times Q_2^{m-t}$ and noise parameter $\gamma > 0$.*
Input: *a Label Cover instance $\mathcal{L} = (U, V, E, L, R, \Pi)$.*
Output: *a weighted OCSP($\mathcal{P}$) instance $\mathcal{I} = (\mathcal{X}, \mathcal{C})$ where $\mathcal{X} = (U \times Q_1^L) \cup (V \times Q_2^R)$. The distribution of constraints in $\mathcal{C}$ is obtained by sampling a random edge $e = (u, v) \in E$ with projection π_e and then $(\mathbf{X}, \mathbf{Y})$ from $\mathcal{T}^{(\gamma)}_{\pi_e}(\mathcal{D})$; the constraint is the predicate $\mathcal{P}$ applied on the tuple*

$$((u, \mathbf{x}^{(1)}), \ldots, (u, \mathbf{x}^{(t)}), (v, \mathbf{y}^{(t+1)}), \ldots, (v, \mathbf{y}^{(m)})).$$

An assignment to $\mathcal{I}$ is seen as a collection of functions, $\{f_u\}_{u \in U} \cup \{g_v\}_{v \in V}$, where $f_u : Q_1^L \to \mathbb{Z}$ and $g_v : Q_2^R \to \mathbb{Z}$. The value of an assignment is:

$$\mathop{\mathbf{E}}_{\substack{\mathbf{e}=(u,v)\in E;\\ (\mathbf{X},\mathbf{Y})\in\mathcal{T}^{(\gamma)}_{\pi_\mathbf{e}}(\mathcal{D})}} \mathcal{P}\left((f_u, g_v) \circ (\mathbf{X}, \mathbf{Y})\right) = \mathop{\mathbf{E}}_{\mathbf{e}=(u,v)\in E}[\mathsf{Acc}_{f_u,g_v}(\mathcal{T}^{(\gamma)}_{\pi_\mathbf{e}}(\mathcal{D}))].$$

Lemma 2 now implies that if $\mathcal{L}$ is satisfiable, then the value of the instance output is also high.

Lemma 3. *If λ is a labeling of $\mathcal{L}$ satisfying a fraction c of its constraints, then the ordering assignment $f_u(\boldsymbol{x}) = x_{\lambda(u)}$, $g_v(\boldsymbol{y}) = y_{\lambda(v)}$ satisfies at least a fraction $c \cdot (\mathbf{E}_{\mathbf{x}\sim\mathcal{D}}[\mathcal{P}(\boldsymbol{\sigma}(\mathbf{x}))] - \gamma m)$ of the constraints of $R^{(\mathcal{P})}_{\mathcal{D},\gamma}(\mathcal{L})$. In particular, there is an ordering of $R^{(\mathcal{P})}_{\mathcal{D},\gamma}(\mathcal{L})$ attaining a value $\mathsf{val}(\mathcal{L}) \cdot (\mathbf{E}_{\mathbf{x}\sim\mathcal{D}}[\mathcal{P}(\boldsymbol{\sigma}(\mathbf{x}))] - \gamma m)$ that is* oblivious *to the distribution $\mathcal{D}$.*

On the other hand, we also extend the decoupling property of the dictatorship test to the instance output if $\mathsf{val}(\mathcal{L})$ is small. This is the technical core of the paper and is proved in Sect. 5.

Theorem 7. *Suppose that $\mathcal{D}$ over $Q_1^t \times Q_2^{m-t}$ satisfies the following properties:*

- *$\mathcal{D}$ has uniform marginals.*
- *For every $i > t$, $\mathcal{D}_i$ is independent of $\mathcal{D}_{\leq t}$.*

Then, for every $\varepsilon > 0$ and $\gamma > 0$ there exists $\varepsilon_{LC} > 0$ such that if $\mathsf{val}(\mathcal{L}) \leq \varepsilon_{LC}$ then for every assignment $A = \{f_u\}_{u \in U} \cup \{g_v\}_{v \in V}$ to $\mathcal{I}$ it holds that

$$\mathsf{val}(A; R^{(\mathcal{P})}_{\mathcal{D},\gamma}(\mathcal{L})) \leq \mathsf{val}(A; R^{(\mathcal{P})}_{\mathcal{D}^\perp,\gamma}(\mathcal{L})) + \varepsilon.$$

In particular, $\mathsf{val}(R^{(\mathcal{P})}_{\mathcal{D},\gamma}(\mathcal{L})) \leq \mathsf{val}(R^{(\mathcal{P})}_{\mathcal{D}^\perp,\gamma}(\mathcal{L})) + \varepsilon$.

4 Applications of the General Result

In this section, we prove the inapproximability of Max BTW and Max NBTW using the general hardness result of Sect. 3. We also prove the hardness of MAS using a gadget reduction from Max NBTW. Due to lack of space, the inapproximability of Max 2t-SO is deferred to the full version.

4.1 Hardness of Maximum Betweenness

For an integer q, define the distribution $\mathcal{D}$ over $\{-1, q\} \times [q] \times [q]$ by picking $\mathbf{x}_1 \sim \{-1, q\}$, $\mathbf{y}_2 \sim [q]$, and setting $\mathbf{y}_3 = \mathbf{y}_2 + 1 \bmod q$ if $\mathbf{x}_1 = q$ and $\mathbf{y}_2 - 1$ otherwise. This distribution has the following properties which can be readily verified.

Proposition 1. *Let* $(\mathbf{x}_1, \mathbf{y}_2, \mathbf{y}_3) \sim \mathcal{D}$. *Then the following holds:*

1. $\mathcal{D}$ *has uniform marginals.*
2. *The marginals* $\mathbf{y}_2$ *and* $\mathbf{y}_3$ *are independent of* $\mathbf{x}$.
3. $(\mathbf{y}_2, \mathbf{y}_3)$ *has the same distribution as* $(\mathbf{y}_3, \mathbf{y}_2)$.
4. $\mathbf{E}_{\mathbf{x}_1,\mathbf{y}_2,\mathbf{y}_3 \sim \mathcal{D}}[\mathrm{BTW}(\mathbf{x}_1, \mathbf{y}_2, \mathbf{y}_3)] \geq 1 - 1/q$.

Let $\mathcal{D}^{\perp}$ be the decoupled distribution of $\mathcal{D}$ which draws the first coordinate independently of the remaining and $\gamma > 0$ a noise parameter. Given a LC instance $\mathcal{L}$ and consider applying Reduction 2 to $\mathcal{L}$ with test distributions $\mathcal{D}$ and $\mathcal{D}^{\perp}$, obtaining Max BTW instances $\mathcal{I} = R^{\mathrm{BTW}}_{\mathcal{D},\gamma}(\mathcal{L})$ and $\mathcal{I}^{\perp} = R^{\mathrm{BTW}}_{\mathcal{D}^{\perp},\gamma}(\mathcal{L})$.

Lemma 4 (Completeness). *If* $\mathsf{val}(\mathcal{L}) = 1$ *then* $\mathsf{val}(\mathcal{I}) \geq 1 - 1/q - 3\gamma$.

Proof. This is an immediate corollary of Lemma 3 and Prop. 1. □

Lemma 5 (Soundness). *For every* $\varepsilon > 0$, $\gamma > 0$, q, *there is an* $\varepsilon_{LC} > 0$ *such that if* $\mathsf{val}(\mathcal{L}) \leq \varepsilon_{LC}$ *then* $\mathsf{val}(\mathcal{I}) \leq 1/2 + \varepsilon$.

Proof. We note that Prop. 1 asserts that $\mathcal{D}$ satisfies the conditions of Theorem 7 and it suffices to show $\mathsf{val}(\mathcal{I}^{\perp}) \leq 1/2$. Let $\{f_u : \{0,1\}^L \to \mathbb{Z}\}_{u \in U}, \{g_v : [q]^R \to \mathbb{Z}\}_{v \in V}$ be an arbitrary assignment to $\mathcal{I}^{\perp}$. Fix an LC edge $\{u, v\}$ with projection π and consider the mean value of constraints produced for this edge by the construction:

$$\mathop{\mathbf{E}}_{\mathbf{x}^{(1)},\mathbf{y}^{(2)},\mathbf{y}^{(3)} \leftarrow \mathcal{T}^{(\gamma)}_{\pi}(\mathcal{D}^{\perp})} \left[\mathrm{BTW}(f_u(\mathbf{x}^{(1)}), g_v(\mathbf{y}^{(2)}), g_v(\mathbf{y}^{(3)})) \right]. \tag{2}$$

As noted in Prop. 1, $(\mathbf{y}^{(2)}, \mathbf{y}^{(3)})$ has the same distribution as $(\mathbf{y}^{(3)}, \mathbf{y}^{(2)})$ when drawn from $\mathcal{D}$. Consequently, when drawing arguments from the decoupled test distribution, the probability of a specific outcome $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(2)}, \boldsymbol{x}^{(3)})$ equals the probability of $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(3)}, \boldsymbol{x}^{(2)})$. For strict orderings, at most one of the two can satisfy the predicate BTW. Thus, the expression in (2), and in effect $\mathsf{val}(\mathcal{I}^{\perp})$, is bounded by $1/2$. □

Theorem 2 is now an immediate corollary of Lemmas 4 and 5, taking $q = \lceil 2/\varepsilon \rceil$ and $\gamma = \varepsilon/6$.

4.2 Hardness of Maximum Non-betweenness

For an implicit parameter q, define a distribution $\mathcal{D}$ over $[q]^3$ by picking $\mathbf{x}_1, \mathbf{x}_2 \sim [q]$ and setting $\mathbf{x}_3 = \mathbf{x}_1 + \mathbf{x}_2 \bmod q$.

Proposition 2. *The distribution $\mathcal{D}$ satisfies the following:*

1. *$\mathcal{D}$ is pairwise independent with uniform marginals,*
2. *and $\mathbf{E}_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3\sim\mathcal{D}}[\mathrm{NBTW}(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)] \geq 1 - 3/q$.*

A straightforward application of the general inapproximability with $t = 1$ shows that $\mathbf{x}_1$ is decoupled from $\mathbf{x}_2$ and $\mathbf{x}_3$ unless $\mathsf{val}(\mathcal{L})$ is large. Further, pairwise independence implies that the decoupled distribution is simply the uniform distribution over $[q]^3$. However, this does not suffice to prove approximation resistance and in fact the value could be greater than 2/3. To see this, note that if $\{f_u\}_{u\in U}, \{g_v\}_{v\in V}$ is an ordering of the instance from the reduction, then the first coordinate of every constraint is a variable of the form $f_u(\cdot)$ while the rest are $g_v(\cdot)$. Thus, ordering the $f_u(\cdot)$ variables in the middle and randomly ordering $g_v(\cdot)$ on both sides satisfies a fraction 3/4 of the constraints.

To remedy this and prove approximation resistance, we permute $\mathcal{D}$ by swapping the last coordinate with each of the coordinates and overlay the instances obtained by the reduction obtained from each of these distributions. For $1 \leq j \leq 3$, define $\mathcal{D}_j$ as the distribution over $[q]^3$ obtained by first sampling from $\mathcal{D}$ and then swapping the jth and third coordinate (in other words, the jth coordinate is the sum of the other two, which are picked independently at random). Similarly, define NBTW_j as the ordering predicate which is true if the j'th argument does not lie between the other two, e.g., $\mathrm{NBTW}_3 = \mathrm{NBTW}$.

As in the previous section, take a LC instance $\mathcal{L}$ and consider applying Reduction 2 to $\mathcal{L}$ with the distributions $\mathcal{D}_j$, and write $\mathcal{I}_j = R^{\mathrm{NBTW}_j}_{\mathcal{D}_j,\gamma}(\mathcal{L})$. Similarly write $\mathcal{I}_j^{\perp} = R^{\mathrm{NBTW}_j}_{\mathcal{D}_j^{\perp},\gamma}(\mathcal{L})$ for the corresponding decoupled instances.

As the distributions $\mathcal{D}_j$ are over the same domain $[q]^3$, the instances $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$ are over the same variables. We define a new instance $\mathcal{I}$ over the same variables as the "sum" $\frac{1}{3}\sum_{j\in[3]} \mathcal{I}_j$, defined by taking all constraints in $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$ (with multiplicities) and then normalizing their weight by 1/3.

Lemma 6 (Completeness). *If $\mathsf{val}(\mathcal{L}) = 1$ then $\mathsf{val}(\mathcal{I}) \geq 1 - 3/q - 3\gamma$.*

Proof. This is an immediate corollary of Lemma 3 and Prop. 2. □

Lemma 7 (Soundness). *For every $\varepsilon > 0$, $\gamma > 0$, q, there is an $\varepsilon_{LC} > 0$ such that if $\mathsf{val}(\mathcal{L}) \leq \varepsilon_{LC}$ then $\mathsf{val}(\mathcal{I}) \leq 2/3 + \varepsilon$.*

Proof. Again our goal is to use Theorem 7 and we start by bounding $\mathsf{val}(\mathcal{I}^{\perp})$. To do this, note that the decoupled distributions $\mathcal{D}_j$ are in fact the uniform distribution over $[q]^3$ and in particular do not depend on j. This means that the distributions of variables to which NBTW_j is applied in $\mathcal{I}_j^{\perp}$ is independent of j, e.g., if $\mathcal{I}_1^{\perp}$ contains the constraint $\mathrm{NBTW}_1(z_1, z_2, z_3)$ with weight w then $\mathcal{I}_2^{\perp}$ contains the constraint $\mathrm{NBTW}_2(z_1, z_2, z_3)$ with the same weight). In other words, $\mathcal{I}^{\perp}$ can be thought of as having constraints of the form $\mathbf{E}_j[\mathrm{NBTW}_j(z_1, z_2, z_3)]$. It is readily verified that $\mathbf{E}_j[\mathrm{NBTW}_j(a, b, c)]] \leq 2/3$ for every a, b, c.

Getting back to the main task – bounding $\mathsf{val}(\mathcal{I})$ – fix an arbitrary assignment $A = \{f_u : [q]^L\}_{u\in U} \cup \{g_v : [q]^R\}_{v\in V}$ of $\mathcal{I}$. By Theorem 7, $\mathsf{val}(A; \mathcal{I}_j) \leq \mathsf{val}(A; \mathcal{I}_j^{\perp}) +$

ε for $j \in [3]$. It follows that $\mathsf{val}(A;\mathcal{I}) \leq \mathsf{val}(A;\mathcal{I}^{\perp}) + \varepsilon$ and therefore, since A was arbitrary, it holds that $\mathsf{val}(\mathcal{I}) \leq \mathsf{val}(\mathcal{I}^{\perp}) + \varepsilon \leq 2/3 + \varepsilon$, as desired. □

4.3 Hardness of Maximum Acyclic Subgraph

The inapproximability of MAS is from a simple gadget reduction from the inapproximability gap of Max NBTW. We claim the following properties of the directed graph shown in Sect. 1. The proof and the routine application of the lemma to derive Theorem 1 are given in the full version.

Lemma 8. *Consider an ordering $\mathcal{O}$ of x, y, z. Then,*

1. *if* $\mathrm{NBTW}(\mathcal{O}(x), \mathcal{O}(y), \mathcal{O}(z)) = 1$, *then* $\max_{\mathcal{O}'} \mathsf{val}(\mathcal{O}'; H) = 5/6$ *where the* max *is over all extensions $\mathcal{O}' : V \to \mathbb{Z}$ of $\mathcal{O}$ to V.*
2. *if* $\mathrm{NBTW}(\mathcal{O}(x), \mathcal{O}(y), \mathcal{O}(z)) = 0$, *then* $\max_{\mathcal{O}'} \mathsf{val}(\mathcal{O}'; H) = 4/6$ *where the* max *is over all extensions of $\mathcal{O}$ to V.*

5 Analysis of the Reduction

In this section we prove Theorem 7 which bounds the value of the instance generated by the reduction in terms of the decoupled distribution. Throughout, we fix an LC instance, $\mathcal{L}$, a predicate $\mathcal{P}$, an OCSP instance $\mathcal{I}$ obtained by the procedure $R^{(\mathcal{P})}_{\mathcal{D},\gamma}$ for a distribution $\mathcal{D}$ and noise-parameter γ. We further assume the set $\{f_u\}_{u \in U} \cup \{g_v\}_{v \in V}$ is the assignment we are interested in analyzing. The proof involves three major steps. First, we show that the functions, which are $\mathbb{Z}$-valued, can be approximated by functions on a finite domain via bucketing (see Sect. 5.1). This approximation makes it amenable to tools developed in the context of analyzing finite-domain CSPs [22,6]; we use these tools in Sect. 5.2 to prove the decoupling property of the dictatorship test. Finally, this decoupling is extended to the reduction hence bounding the value of $\mathcal{I}$.

5.1 Bucketing

For an integer Γ, we approximate the function $f_u : Q_1^L \to \mathbb{Z}$ by partitioning the domain into Γ pieces. Put $q_1 = |Q_1|$ and partition the set Q_1^L into sets $B_1^{(f_u)}, \ldots, B_\Gamma^{(f_u)}$ of size q_1^L/Γ such that if $\boldsymbol{x} \in B_i^{(f_u)}$ and $\boldsymbol{y} \in B_j^{(f_u)}$ for some $i < j$ then $f(\boldsymbol{x}) < f(\boldsymbol{y})$. Note that this is possible as long as the parameter Γ divides q_1^L which will be the case. Let $F_u : Q_1^L \to [\Gamma]$ specify the mapping of points to the bucket containing it, and $F_u^{(a)} : Q_1^L \to \{0,1\}$ the indicator of points assigned to $B_a^{(f_u)}$. Partition $g_v : Q_2^R \to \mathbb{Z}$ similarly into buckets $\{B_a^{(g_v)}\}$ obtaining $G_v : Q_2^R \to [\Gamma]$ and $G_v^{(a)} : Q_2^R \to \{0,1\}$.

Now we show that the acceptance probability of the dictatorship test – see (1) in Sect. 3 – applied to an edge $e = (u,v)$ of the LC instance $\mathcal{L}$ can be approximated by a bucketed version. Fix an edge $e = (u,v)$ and put $f = f_u$, $g = g_v$. As before, we denote a query tuple, $(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t)}, \mathbf{y}^{(t+1)}, \ldots, \mathbf{y}^{(m)})$

concisely as $(\mathbf{X}, \mathbf{Y})$. Define the bucketed payoff function with respect to f and g, $\wp^{(f,g)} : [\Gamma]^m \to [0,1]$ as:

$$\wp^{(f,g)}(a_1, \ldots, a_m) = \mathop{\mathbf{E}}_{\substack{\mathbf{x}^{(i)} \leftarrow B_{a_i}^{(f)}; i \le t \\ \mathbf{y}^{(j)} \leftarrow B_{a_j}^{(g)}; t < j}} [\mathcal{P}((f,g) \circ (\mathbf{X}, \mathbf{Y}))]$$

and the *bucketed* acceptance probability,

$$\mathsf{BAcc}_{f,g}(\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})) = \mathop{\mathbf{E}}_{(\mathbf{X},\mathbf{Y}) \leftarrow \mathcal{T}_\pi^{(\gamma)}(\mathcal{D})} \left[\wp^{(f,g)}((F,G) \circ (\mathbf{X}, \mathbf{Y})) \right].$$

In other words, bucketing corresponds to generating a tuple $\boldsymbol{a} = (f,g) \circ (\mathbf{X}, \mathbf{Y})$ and replacing each coordinate a_i with a random value from the bucket a_i fell in. We show that above is close to the true acceptance probability $\mathsf{Acc}_{f,g}(\mathcal{T}_\pi^{(\gamma)}(\mathcal{D}))$.

Theorem 8. *For every predicate $\mathcal{P}$, every distribution $\mathcal{D}$ with uniform marginals, every pair of orderings $f : Q_1^L \to \mathbb{Z}$ and $g : Q_2^R \to \mathbb{Z}$, every $\gamma > 0$, projection $\pi : R \to L$, and every Γ,*

$$\left| \mathsf{Acc}_{f,g}(\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})) - \mathsf{BAcc}_{f,g}(\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})) \right| \le m^2 \Gamma^{-\delta},$$

for some $\delta = \delta(\gamma, Q) > 0$ with $Q = \max\{|Q_1|, |Q_2|\}$.

To prove this, we show that f and g have few overlapping pairs of buckets and that the probability of hitting any particular pair is small. Let $R_a^{(f)}$ be the smallest interval in $\mathbb{Z}$ containing $B_a^{(f)}$; and similarly for $R_a^{(g)}$.

Lemma 9 (Few Buckets Overlap). *For every integer Γ there are at most 2Γ choices of pairs $(a,b) \in [\Gamma] \times [\Gamma]$ such that $R_a^{(f)} \cap R_b^{(g)} \neq \emptyset$.*

Proof. Construct the bipartite intersection graph $G_I = (U_I, V_I, E_I)$ where the vertex sets are disjoint copies of $[\Gamma]$, and there is an edge between $a \in U_I$ and $b \in V_I$ iff $R_a^{(f)} \cap R_b^{(g)} \neq \emptyset$. By construction of the buckets, the graph does not contain any pair of distinct edges $(u,v), (u',v')$ such that $u < v$ and $u' > v'$. Consequently, a vertex can have at most two neighbors with degree greater than one. Let A be the set of degree-one vertices. Then, $\Delta(G_I[\overline{A}]) \le 2$ and $|E[G_I[\overline{A}]]| \le |U_I + V_I - A|$ while $|E(A, U_I + V_I)| \le |A|$. Since $|E_I| = |E[G_I[\overline{A}]]| + |E(A, U_I + V_I)| \le |U_I + V_I - A| + |A| \le 2\Gamma$. □

Next, we prove a bound on the probability that a fixed pair of the m queries fall in a fixed pair of buckets. For a distribution D over $Q_1^L \times Q_2^R$, define $D^{(\gamma)}$ as the distribution that samples from D and for each of the $|L| + |R|$ coordinates independently with probability γ replaces it with a new sample from D. $D^{(\gamma)}$ is representative of the projection of $\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})$ to two specific coordinates and we show that noise prevents the buckets from correlating well.

Lemma 10. *Let D be a distribution over $Q_1^L \times Q_2^R$ whose marginals are uniform in Q_1^L and Q_2^R and $D^{(\gamma)}$ be as defined above. For every integer Γ and every pair of functions $F : Q_1^L \to \{0,1\}$ and $G : Q_2^R \to \{0,1\}$ such that $\mathbf{E}[F(\mathbf{x})] = \mathbf{E}[G(\mathbf{y})] = 1/\Gamma$,*

$$\mathop{\mathbf{E}}_{(\mathbf{x},\mathbf{y})\in D^{(\gamma)}}[F(\mathbf{x})G(\mathbf{y})] \leq \Gamma^{-(1+\delta)}$$

for some $\delta = \delta(\gamma, Q) > 0$ where $Q = \min\{|Q_1|, |Q_2|\}$.

Proof. Without loss of generality, let $|Q_1| = \min\{|Q_1|, |Q_2|\}$. Set $q = 2 + \delta' > 2$ as in Lemma 1, $1/q' = 1 - 1/q$, and define $H(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{E}_{\mathbf{y}|\mathbf{x}}[T_{1-\gamma}G(\mathbf{y})]$. Then,

$$\begin{aligned}\mathop{\mathbf{E}}_{(\mathbf{x},\mathbf{y})\in D^{(\gamma)}}[F(\mathbf{x})G(\mathbf{y})] &= \mathop{\mathbf{E}}_{\mathbf{x}}[T_{1-\gamma}F(\mathbf{x})H(\mathbf{x})] \leq \|T_{1-\gamma}F\|_q \|H\|_{q'} \\ &\leq \|F\|_2 \|H\|_{q'} = \|F\|_2 \|T_{1-\gamma}G\|_{q'} \\ &\leq \|T_{1-\gamma}F\|_q \|G\|_{q'} = \Gamma^{-(1/2+1/q')} = \Gamma^{-(1+\delta'/2(2+\delta'))},\end{aligned}$$

using Lemma 1, convexity of norms, and the contractivity of $T_{1-\gamma}$. □

Note that the above lemma applies to queries to the same function as well, setting $F = G$, etc. To complete the proof of Theorem 8, we apply the above lemma to every distinct pair of the m queries made in $\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})$, bounding the difference between the true acceptance probability and the bucketed version.

Proof (of Theorem 8). Note that the bucketed payoff $\wp^{(f,g)}((F,G) \circ (\mathbf{X},\mathbf{Y}))$ is equal to the true payoff $\mathcal{P}((f,g) \circ (\mathbf{X},\mathbf{Y}))$ except possibly when two pairs of outputs fall in an overlapping pair of buckets. Hence it suffices to bound the probability of this happening by $m^2\Gamma^{-\delta}/2$.

Fix a pair of inputs, say $x^{(i)}$ and $y^{(j)}$; the argument is the same if we choose two x inputs or two y inputs. Let $a = F(x^{(i)})$ and $b = G(y^{(j)})$. By Lemma 9 there are at most 2Γ possible values (a,b) such that the buckets indexed by a and b are overlapping. From Lemma 10, the probability that $F(\mathbf{x}^{(i)}) = a$ and $G(\mathbf{y}^{(j)}) = b$ is at most $\Gamma^{-1-\delta}$. By a union bound, the two outputs $F(\mathbf{x}^{(i)})$, $G(\mathbf{y}^{(j)})$ consequently fall in overlapping buckets with probability at most $2\Gamma^{-\delta}$. As there are at most $\binom{m}{2} \leq m^2/2$ pairs of outputs, the proof is complete. □

5.2 Soundness of the Dictatorship Test

We now reap the benefits of bucketing and prove the decoupling property of the dictatorship test alluded to in Sect. 3.

Lemma 11. *For every predicate $\mathcal{P}$ and distribution $\mathcal{D}$ satisfying the conditions of Theorem 7, and any noise rate $\gamma > 0$, projection $\pi : R \to L$, and bucketing parameter Γ, the following holds. For any functions $f : Q_1^L \to \mathbb{Z}$, $g : Q_2^R \to \mathbb{Z}$ with bucketing functions $F : Q_1^L \to [\Gamma]$, $G : Q_2^R \to [\Gamma]$ it holds that*

$$\begin{aligned}&\left|\mathsf{Acc}_{f,g}(\mathcal{T}_\pi^{(\gamma)}(\mathcal{D})) - \mathsf{Acc}_{f,g}(\mathcal{T}_\pi^{(\gamma)}(\mathcal{D}^\perp))\right| \\ &\qquad \leq \gamma^{-1/2} m^{1/2} 4^m \Gamma^m \sum_{a,b\in[\Gamma]} \mathrm{CrInf}_\pi^{(1-\gamma)}\left(F^{(a)}, G^{(b)}\right)^{1/2} + 2\Gamma^{-\delta}m^2.\end{aligned}$$

Due to space limitations, the proof of Lemma 11 is deferred to the full version. Roughly, the idea is to prove a similar bound for the bucketed acceptance probability $\mathsf{BAcc}_{f,g}(\mathcal{T}_\pi^{(\gamma)}(\mathcal{D}))$ and then use the Theorem 8. The bound for the bucketed value goes via the invariance principle and uses a few sophisticated but standard estimates developed in the works of, amongst others, Mossel [17] and Wenner [22]. With Lemma 11 in place, Theorem 7 can be derived using standard influence-decoding techniques; this is also deferred to the full version.

6 Conclusion

We gave improved inapproximability for several important OCSPs. Our characterization is by no means complete and leave behind several interesting open problems. Closing the gap in the approximability of MAS is wide open and probably no easier than resolving the approximability for Max Cut and other 2-CSPs. In particular, getting any factor close to $1/2$ seems to require new ideas. Max BTW has an approximation algorithm that satisfies half of the constraints if all the constraints can be simultaneously satisfied. Thus improving our result to obtaining perfect completeness is particularly enticing. Finally, improving our general hardness result to only requiring that $\mathcal{D}$ is pairwise independent is interesting especially in light of the analogous results for CSPs [4,6].

Acknowledgement. We would like to thank Johan Håstad for suggesting this problem and for numerous helpful discussions regarding the same. We acknowledge ERC Advanced Grant 226203 and Swedish Research Council Grant 621-2012-4546 for making this project feasible.

References

1. Arora, S., Barak, B., Steurer, D.: Subexponential algorithms for Unique Games and related problems. In: FOCS, pp. 563–572. IEEE Computer Society (2010)
2. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM 45(3), 501–555 (1998)
3. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. J. ACM 45(1), 70–122 (1998)
4. Austrin, P., Mossel, E.: Approximation resistant predicates from pairwise independence. CCC 18(2), 249–271 (2009)
5. Barak, B., Brandão, F.G.S.L., Harrow, A.W., Kelner, J.A., Steurer, D., Zhou, Y.: Hypercontractivity, sum-of-squares proofs, and their applications. In: Karloff, H.J., Pitassi, T. (eds.) STOC, pp. 307–326. ACM (2012)
6. Chan, S.O.: Approximation resistance from pairwise independent subgroups. In: STOC, pp. 325–337 (2013)
7. Charikar, M., Guruswami, V., Manokaran, R.: Every permutation CSP of arity 3 is approximation resistant. In: CCC, pp. 62–73 (2009)
8. Charikar, M., Makarychev, K., Makarychev, Y.: On the advantage over random for Maximum Acyclic Subgraph. In: FOCS, pp. 625–633. IEEE Computer Society (2007)

9. Chor, B., Sudan, M.: A geometric approach to betweenness. SIAM J. Disc. Math. 11(4), 511–523 (1998)
10. Engebretsen, L., Holmerin, J.: More efficient queries in PCPs for NP and improved approximation hardness of maximum CSP. Rand. Struct. Algo. 33(4), 497–514 (2008)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
12. Guruswami, V., Håstad, J., Manokaran, R., Raghavendra, P., Charikar, M.: Beating the random ordering is hard: Every ordering CSP is approximation resistant. SIAM J. Comput. 40(3), 878–914 (2011)
13. Guruswami, V., Manokaran, R., Raghavendra, P.: Beating the random ordering is hard: Inapproximability of Maximum Acyclic Subgraph. In: FOCS, pp. 573–582 (2008)
14. Guruswami, V., Raghavendra, P., Saket, R., Wu, Y.: Bypassing UGC from some optimal geometric inapproximability results. In: Rabani, Y. (ed.) SODA, pp. 699–717. SIAM (2012)
15. Håstad, J.: Some optimal inapproximability results. J. ACM 48(4), 798–859 (2001)
16. Khot, S.: On the power of unique 2-prover 1-round games. In: STOC, pp. 767–775 (2002)
17. Mossel, E.: Gaussian bounds for noise correlation of functions. Geo. and Func. Anal. 19 (2010)
18. Newman, A.: The Maximum Acyclic Subgraph Problem and degree-3 graphs. In: Goemans, M.X., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX-RANDOM 2001. LNCS, vol. 2129, pp. 147–158. Springer, Heidelberg (2001)
19. Raz, R.: A parallel repetition theorem. SIAM J. Comput. 27(3), 763–803 (1998)
20. Samorodnitsky, A., Trevisan, L.: A PCP characterization of NP with optimal amortized query complexity. In: Yao, F.F., Luks, E.M. (eds.) STOC, pp. 191–199. ACM (2000)
21. Trevisan, L., Sorkin, G.B., Sudan, M., Williamson, D.P.: Gadgets, approximation, and linear programming. SIAM J. Comput. 29(6), 2074–2097 (2000)
22. Wenner, C.: Circumventing d-to-1 for approximation resistance of satisfiable predicates strictly containing parity of width four (extended abstract). In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) APPROX/RANDOM 2012 LNCS, vol. 7408, pp. 325–337. Springer, Heidelberg (2012)
23. Wolff, P.: Hypercontractivity of simple random variables. Studia Mathematica, 219–326 (2007)

Approximating Large Frequency Moments with Pick-and-Drop Sampling

Vladimir Braverman[1,⋆] and Rafail Ostrovsky[2,⋆⋆]

[1] Johns Hopkins University, Department of Computer Science
vova@cs.jhu.edu
[2] University of California Los Angeles,
Department of Computer Science and Department of Mathematics
rafail@cs.ucla.edu

Abstract. Given data stream $D = \{p_1, p_2, \ldots, p_m\}$ of size m of numbers from $\{1, \ldots, n\}$, the frequency of i is defined as $f_i = |\{j : p_j = i\}|$. The k-th *frequency moment* of D is defined as $F_k = \sum_{i=1}^{n} f_i^k$. We consider the problem of approximating frequency moments in insertion-only streams for $k \geq 3$. For any constant c we show an $O(n^{1-2/k} \log(n) \log^{(c)}(n))$ upper bound on the space complexity of the problem. Here $\log^{(c)}(n)$ is the iterative log function. Our main technical contribution is a non-uniform sampling method on matrices. We call our method a *pick-and-drop sampling*; it samples a heavy element (i.e., element i with frequency $\Omega(F_k)$) with probability $\Omega(1/n^{1-2/k})$ and gives approximation $\tilde{f}_i \geq (1-\epsilon)f_i$. In addition, the estimations never exceed the real values, that is $\tilde{f}_j \leq f_j$ for all j. For constant ϵ, we reduce the space complexity of finding a heavy element to $O(n^{1-2/k} \log(n))$ bits. We apply our method of recursive sketches and resolve the problem with $O(n^{1-2/k} \log(n) \log^{(c)}(n))$ bits. We reduce the ratio between the upper and lower bounds from $O(\log^2(n))$ to $O(\log(n) \log^{(c)}(n))$. Thus, we provide a (roughly) quadratic improvement of the result of Andoni, Krauthgamer and Onak (FOCS 2011).

Keywords: Data streams, frequency moments, sampling.

1 Introduction

Given a sequence $D = \{p_1, p_2, \ldots, p_m\}$ of size m of numbers from $\{1, \ldots, n\}$, a frequency of i is defined as

$$f_i = |\{j : p_j = i\}|. \tag{1}$$

⋆ This work was supported in part by DARPA grant N660001-1-2-4014. Its contents are solely the responsibility of the author and do not represent the official view of DARPA or the Department of Defense.

⋆⋆ Research supported in part by NSF grants CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 42–57, 2013.

The k-th *frequency moment* of D is defined as

$$F_k = \sum_{i=1}^{n} f_i^k. \tag{2}$$

The problem of approximating frequency moments in one pass over D and using *sublinear* space has been introduced in the award-winning paper of Alon, Matias and Szegedy [1]. In particular, they observed a striking difference between "small" and "large" values of k: it is possible to approximate $F_k, k \leq 2$ in polylogarithmic space, but polynomial space is required when $k > 2$. Since 1996, approximating F_k has become one of the most inspiring problems in the theory of data streams. The incomplete list of papers on frequency moments include [24, 19, 4, 13, 5, 25, 16–18, 20, 23, 9, 30, 10, 28, 29, 32, 34, 6, 12, 26, 2, 21, 22, 36, 27, 3, 7] and references therein. We omit the detailed history of the problem and refer a reader to [31, 35] for overviews.

In this paper we consider the case when $k \geq 3$. In their breakthrough paper Indyk and Woodruff [25] gave the first solution that is optimal up to a polylogarithmic factor. Numerous improvements were proposed in the later years (see the references above) and the latest bounds are due to Andoni, Krauthgamer and Onak [2] (and concurrently Braverman and Ostrovsky [10]) and Ganguly [21]. The latest bound by Ganguly [21] is

$$O(k^2\epsilon^{-2}n^{1-2/k}E(p,n)\log(n)\log(nmM)/\min(\log(n),\epsilon^{4/k-2}))$$

where, $E(k,n) = (1-2/k)^{-1}(1-n^{-4(1-2/k)})$. This bound is roughly $O(n^{1-2/k}\log^2(n))$ for constant ϵ, k. The best known lower bound for insertion-only streams is $\Omega(n^{1-2/k})$, due to Chakrabarti, Khot and Sun [13].

We consider the problem of approximating frequency moments in insertion-only streams for $k \geq 3$. For any constant c we show an $O(n^{1-2/k}\log(n)\log^{(c)}(n))$ upper bound on the space complexity of the problem. Here $\log^{(c)}(n)$ is the iterative $\log$ function. To simplify the presentation, we make the following assumptions: n and m are polynomially far[1]; approximation error ϵ and parameter k are constants. We observe a natural bijection between streams and special matrices[2]. Our main technical contribution is a non-uniform sampling method on matrices that are accessed in a row-by-row way. We call our method a *pick-and-drop sampling*; it samples a heavy element (i.e., element i with frequency $\Omega(F_k)$) with probability $\Omega(1/n^{1-2/k})$ and gives approximation $\tilde{f}_i \geq (1-\epsilon)f_i$. In addition, the estimations never exceed the real values, that is $\tilde{f}_j \leq f_j$ for all j. As a result, we reduce the space complexity of finding a heavy element to $O(n^{1-2/k}\log(n))$ bits. To reduce the problem of the F_k to the problem of finding heavy element we use our method of recursive sketches [8]. As a result, we resolve the problem with $O(n^{1-2/k}\log(n)\log^{(c)}(n))$ bits. We do not try to optimize the space complexity as a function of ϵ.

[1] Our proofs are correct even without this assumption but become longer. If we drop this assumption then the factor $\log(n)$ in our final bound should be replaced with $O(\log(nm))$.

[2] We stress that the bijection is for the presentation purposes only. We do not address problems related to linear algebra such as [15].

1.1 Pick-and-Drop Sampling

Pick-and-drop sampling has been inspired by a very natural behavior of children. We observed the following pattern: a child picks a toy, briefly plays with it, then drops the toy and picks a new one. This pattern is repeated until the child picks the favorite toy and keeps it for a long time. Indeed, children develop algorithms for selectivity [33].

To illustrate the pick-and-drop method by example, assume that $m = r * t$ where $r = \lceil n^{1/k} \rceil$ and consider $r \times t$ matrix M with entries $m_{i,j} = p_{t(i-1)+j}$. For $m \leq n$ we aim to solve the following promise problem with probability $2/3$:

- Case 1: all frequencies are either zero or one.
- Case 2: z appears in every row of M exactly once (thus $f_z = r$). All other frequencies are either zero or one.

Consider the following sampling method. Pick r i.i.d. random numbers $I_1, \ldots, I_r$, where I_i is uniformly distributed on $\{1, 2, \ldots, t\}$. For each $i = 1 \ldots r - 1$ we check if there is a duplicate of m_{i,I_i} in the row $i + 1$. If the duplicate is found then we output "Case 2" and stop; otherwise we repeat the test for $i + 1$. That is, the i-th sample is "dropped," and the $(i + 1)$-th sample is "picked". We repeat this experiment T times independently and output "Case 1" if no duplicate is found. Note that if the input represents Case 1 then our method will always output "Case 1." Consider Case 2 and observe that if $m_{i,I_i} = z$ then our method will output "Case 2". Indeed, since z appears in every row, the duplicate of z will be found. Note that

$$P(\cap_{i=1}^{r}(z \neq m_{i,I_i})) = \left(1 - \frac{1}{t}\right)^{rT}. \tag{3}$$

Recall that $m \leq n, m = rt, r = \lceil n^{1/k} \rceil$. If $T = O(n^{1-2/k})$ with sufficiently large constant then the probability of error (3) is smaller than $1/3$. We conclude that our promise problem can be resolved with $O(n^{1-2/k} \log(n))$ space. Note how our solution depends on r. In general, the matrix should be carefully chosen.

Unfortunately the distribution of the frequent element in the stream can be arbitrary. Also our algorithm must recognize "noisy" frequencies that are large but negligible. Clearly, the sampling must be more intricate but, luckily, not by much. In particular, the following method works. We introduce a local counter for each sample that counts the number of times m_{i,I_i} appears in the suffix of the i-th row (this counting method is used in [1] for the entire stream). We maintain a global sample (and a global counter) as functions of the local samples and counters. Initially the global sample is the local sample of the first row. Under certain conditions, the global sample can be "dropped." If this is the case then the local sample of the current row is "picked" and becomes the new global sample. The global sample is "dropped" when the local counter exceeds the global one. Also, the global sample is dropped if the global counter does not grow fast enough. We use function λq where λ is a parameter and q is the number of rows that the global counter survived. If the global counter is smaller than λq then the global sample is "dropped."

In our analysis we concentrate on the case when 1 is the heavy element, but it is possible to repeat our arguments for any i. Our main technical contribution is Theorem

1 that claims that 1 will be outputted with probability $\Omega(\frac{f_1}{t})$ for sufficiently large f_1. Interestingly, Theorem 1 holds for arbitrary distributions of frequencies. In Theorem 2 we show that there exist r, t, λ such that a bound similar to (3) holds. We combine our new method with [8] and obtain our main result in Theorem 4.

Our pick-and-drop sampling has been inspired by the first sublinear method for the large frequency moments: the sampling algorithm from the award-winning paper of Alon, Matias and Szegedy [1]. Specifically, the sampling from [1] can be seen as the sampling on the matrix with a single row, thus omitting the "dropping" steps. As a result we improve the sampling of Alon, Matias and Szegedy by the factor of roughly $\Omega(n^{1/k})$ and generalize their result.

2 Pick-and-Drop Sampling

Let M be a matrix with r rows and t columns and with entries $m_{i,j} \in [n]$. For $i \in [r], j \in [t], l \in [n]$ define:

$$d_{i,j} = |\{j' : j \leq j' \leq t, m_{i,j'} = m_{i,j}\}|, \tag{4}$$

$$f_{l,i} = |\{j \in [t] : m_{i,j} = l\}|, \tag{5}$$

$$f_l = |\{(i,j) : m_{i,j} = l\}|, \tag{6}$$

$$F_k = \sum_{l=1}^{n} f_l^k, G_k = F_k - f_1^k. \tag{7}$$

Note that there is a bijection between $r \times t$ matrices M and streams D of size $r \times t$ with elements $p_{i(t-1)+j} = m_{i,j}$ where the definitions (2), (1) and (6), (7) define equivalent frequency vectors for a matrix and the corresponding stream. W.l.o.g, we will consider streams of size $r \times t$ for some r, t and will interchange the notions of a stream and its corresponding matrix.

Let $\{I_j\}_{j=1}^{r}$ be i.i.d. random variables with uniform distribution on $[t]$. Define for $i = 1, \ldots, r$:

$$s_i = m_{i,I_i}, c_i = d_{i,I_i} \tag{8}$$

Let λ be a parameter. Define the following recurrent random variables:

$$S_1 = s_1, C_1 = c_1, q_1 = 1. \tag{9}$$

Also (for $i = 2, \ldots r$) if

$$(C_{i-1} < \max\{\lambda q_{i-1}, c_i\}) \tag{10}$$

then define

$$S_i = s_i, C_i = c_i, q_i = 1; \tag{11}$$

otherwise, define

$$S_i = S_{i-1}, C_i = C_{i-1} + f_{S_i,i}, q_i = q_{i-1} + 1 \tag{12}$$

Theorem 1. *Let M be a $r \times t$ matrix. Then there exist absolute constants*[3] $\alpha \geq 2 * 10^4, \beta \leq 0.5 * 10^{-4}$ *such that*

$$\alpha(\lambda r + \frac{G_3}{\lambda t} + \frac{G_2}{t}) \leq f_1 \leq \beta t \tag{13}$$

then

$$P(S_r = 1) \geq \frac{f_1}{2t}. \tag{14}$$

2.1 Proof of Theorem 1

Proof. Denote $Q = \{(i,j) : m_{i,j} = 1\}$. For $(i,j) \in Q$ define

$$T_{i,j} = \overline{(A_{i,j} \cup B_{i,j} \cup H_{i,j})}, \tag{15}$$

where for $i > 1$:

$$A_{i,j} = ((C_{i-1} \geq d_{i,j}) \cap (S_{i-1} \neq 1)), \tag{16}$$

for $i < r$:

$$B_{i,j} = \left(\bigcup_{h=i+1}^{r} \left(d_{i,j} + \sum_{u=i+1}^{h-1} f_{1,u} < c_h \right) \right), \tag{17}$$

$$H_{i,j} = \left(\bigcup_{h=i+1}^{r} \left(d_{i,j} + \sum_{u=i+1}^{h-1} f_{1,u} < (h-i)\lambda \right) \right), \tag{18}$$

and $A_{1,j} = B_{r,j} = H_{r,j} = \emptyset$. We have

$$((s_i = 1) \cap (S_{i-1} \neq 1) \cap \overline{A_{i,I_i}}) \subseteq ((s_i = 1) \cap (C_{i-1} < c_i)) \subseteq$$

$$((S_i = 1) \cap (q_i = 1)). \tag{19}$$

Consider the case when $S_i = 1$ and $q_i = 1$ and

$$d_{i,I_i} + \sum_{u=i+1}^{h-1} f_{1,u} \geq \max(\lambda(h-i), c_h)$$

for all $h > i$. In this case S_h will be defined by (12) and not by (11); in particular, $S_h = S_i = 1$. Therefore,

$$((S_i = 1) \cap (q_i = 1) \cap \overline{B_{i,I_i}} \cap \overline{H_{i,I_i}}) \subseteq (\bigcap_{h=i}^{r} (S_h = 1)). \tag{20}$$

Define $V_1 = ((s_1 = 1) \cap T_{1,I_1})$ and, for $i > 1$, $V_i = ((s_i = 1) \cap (S_{i-1} \neq 1) \cap T_{i,I_i})$. If follows from (19), (20) that, for any $i \in [r]$:

$$V_i \subseteq (S_r = 1), \tag{21}$$

[3] We did not try to optimize the constants.

$$V_i \cap V_j = \emptyset. \tag{22}$$

Thus,

$$\sum_{i=1}^{r} P(V_i) = P\left(\cup_{i=1}^{r} V_i\right) \leq P(S_r = 1). \tag{23}$$

For any $i > 1$:

$$P(V_i) \geq P((s_i = 1) \cap T_{i,I_i}) - P(s_i = S_{i-1} = 1).$$

Also,

$$\sum_{i=2}^{r} P(s_i = S_{i-1} = 1) \leq \sum_{i=2}^{r} P((s_i = 1) \cap (\cup_{h \neq i}(s_h = 1))) \leq$$

$$(\sum_{i=1}^{r} P(s_i = 1))^2 = \left(\frac{f_1}{t}\right)^2.$$

For any fixed $(i, j) \in Q$ events $I_i = j$ and $T_{i,j}$ are independent. Indeed, $A_{i,j}$ is defined by $\{S_{i-1}, C_{i-1}\}$ that, in turn, is defined by $\{I_1, \ldots, I_{i-1}\}$. Similarly, $B_{i,j}$ is defined by $\{I_{i+1}, \ldots, I_r\}$. Note that $H_{i,j}$ is a deterministic event. By definition, $\{I_1, \ldots, I_{i-1}, I_{i+1}, \ldots, I_r\}$ are independent of I_i; thus event $I_i = j$ and $T_{i,j} = \overline{(A_{i,j} \cup B_{i,j} \cup H_{i,j})}$ are independent. Thus,

$$\sum_{i=1}^{r} P((s_i = 1) \cap T_{i,I_i}) = \sum_{(i,j) \in Q} P((I_i = j) \cap T_{i,j}) =$$

$$\sum_{(i,j) \in Q} P(I_i = j) P(T_{i,j}) = \frac{1}{t} \sum_{(i,j) \in Q} P(T_{i,j}). \tag{24}$$

Thus,

$$P(S_r = 1) \geq \frac{1}{t} \sum_{(i,j) \in Q} P(T_{i,j}) - \left(\frac{f_1}{t}\right)^2.$$

Lemma 1 implies that $\sum_{(i,j) \in Q} P(T_{i,j}) \geq 0.8 f_1$. Thus if $\beta < 0.3$ then:

$$P(S_r = 1) \geq \frac{f_1}{t}(0.8 - \frac{f_1}{t}) \geq \frac{f_1}{2t}.$$

Here we only use the second part of (13). The first part is used in the proof of Lemma 1.

2.2 Technical Lemmas

Lemma 1. *There exist absolute constants α, β such that (13) implies*

$$\sum_{(i,j) \in Q} P(T_{i,j}) > 0.8 f_1.$$

It follows from Lemmas 7, 14, 12 and the union bound that there exists at least $0.97 f_1$ pairs $(i, j) \in Q$ such that $P(A_{i,j} \cup B_{i,j} \cup H_{i,j}) \leq 0.02$. Recall that $T_{i,j} = \overline{(A_{i,j} \cup B_{i,j} \cup H_{i,j})}$; the lemma follows.

Events of type A For $(i,j) \in Q$ s.t. $i > 1$ and for $l > 1$ define:

$$Y_{l,(i,j)} = \mathbf{1}_{A_{i,j}} \mathbf{1}_{(S_{i-1}=l)},$$

$$Y_{l,i} = \sum_{j \in [t], (i,j) \in Q} Y_{l,(i,j)},$$

$$Y_l = \sum_{i=2}^{r} Y_{l,i},$$

$$Y = \sum_{l=2}^{n} Y_l,$$

Fact 2 *$C_i \le f_{S_i}$. Also, if $q_i = 1$ then $C_i \le f_{S_i,i}$.*

Proof. Follows directly from (11), (12). It is sufficient to prove that, for any i, there exists a set Q_i such that $C_i = |Q_i|$ and simultaneously Q_i is a subset of $\{(i',j) : m_{i',j} = S_i, i' \le i\}$. We prove the above claim by induction on i. For $i = 1$ the claim is true since we can define $Q_1 = \{(1,j) : j \ge I_1\}$. For $i > 2$ the description of the algorithm implies the following. If $q_i = 1$ then we can put $Q_i = \{(i,j) : j \ge I_i\}$. If $q_i > 1$ then define $Q_i = Q_{i-1} \cup \{(i,j) : m_{i,j} = S_i\}$. Note that in this case $S_i = S_{i-1}$. The second part follows from the description of the algorithms: if $p_i = 1$ then $C_i = c_i, S_i = s_i$ and $c_i = d_{i,I_i}(s_i) \le f_{s_i,i}$.

Fact 3

1. $Y_{l,i} \le f_l$,
2. *If* $q_{i-1} = 1$ *then* $Y_{l,i} \le f_{l,i-1}$.

Proof. Let $(i,j) \in Q$ be such that $d_{i,j} > f_l$; then:

$$Y_{l,(i,j)} = \mathbf{1}_{(C_{i-1} \ge d_{i,j})} \mathbf{1}_{(S_{i-1}=l)} = \mathbf{1}_{(f_l \ge C_{i-1})} \mathbf{1}_{(C_{i-1} \ge d_{i,j})} \mathbf{1}_{(S_{i-1}=l)}.$$

We use Fact 2 for the last equality. Thus, $Y_{l,(i,j)} = 0$. Definition of $d_{i,j}$ implies $|\{j : (i,j) \in Q, d_{i,j} \le f_l\}| \le f_l$ for any fixed i and l. Thus,

$$Y_{l,i} = \sum_{j \in [t], (i,j) \in Q} Y_{l,(i,j)} \le f_l.$$

Part 2 following by repeating the above arguments and using the second statement of Fact 2.

Definition 1. *Let $1 \le r_1 \le r_2 \le r$ and $l \in [n]$. Call a pair $[r_1, r_2]$ an l-epoch if*

$$\forall i = r_1, \ldots, r_2 : \quad S_i = l,$$

and

$$q_{r_1} = q_{r_2+1} = 1,$$

and

$$\forall i = r_1 + 1, \ldots, r_2 : \quad q_i = q_{i-1} + 1.$$

Lemma 4. *Let $[r_1, r_2]$ be an l-epoch. If $r_2 > r_1$ then*

$$r_2 - r_1 \leq \frac{1}{\lambda} \sum_{i=r_1}^{r_2-1} f_{l,i}.$$

Proof. First, observe that $q_{r_2-1} = r_2 - r_1$. Second, $q_i > 1$ implies that S_i is defined by (12) and not by (11) for all $r_1 < i \leq r_2$. In particular, $C_{r_1} \leq f_{l,r_1}$ and for $r_1 < i \leq r_2$ we have $C_i = C_{i-1} + f_{l,i}$. Thus,

$$C_{r_2-1} \leq \sum_{i=r_1}^{r_2-1} f_{l,i}.$$

Third, $C_{r_2-1} \geq \lambda q_{r_2-1}$ since (10) must be false for $i = r_2$. Therefore,

$$r_2 - r_1 = q_{r_2-1} \leq \frac{1}{\lambda} C_{r_2-1} \leq \frac{1}{\lambda} \sum_{i=r_1}^{r_2-1} f_{l,i}.$$

Lemma 5. $Y_l \leq \frac{f_l^2}{\lambda} + f_l$.

Proof. Observe that the set $\{i : S_i = l\}$ is a collection of disjoint l-epochs. Recall that $Y_l = \sum_{i=2}^{r} Y_{l,i}$ and $Y_{l,i}$ is non-zero only if S_{i-1} is equal to l. Thus we can rewrite Y_l as:

$$Y_l = \sum_{(r_1,r_2)\text{is an } l\text{-epoch}} \left(\sum_{i=r_1+1}^{r_2+1} Y_{l,i} \right).$$

For any epoch such that $r_2 > r_1$ we have by Lemmas 3 and 4:

$$\sum_{i=r_1+1}^{r_2} Y_{l,i} \leq (r_2 - r_1) f_l \leq \frac{f_l}{\lambda} \sum_{i=r_1}^{r_2-1} f_{l,i}.$$

Since all epochs are disjoint we have

$$Y_l = \sum_{(r_1<r_2)\text{is an } l\text{-epoch}} \left(\sum_{i=r_1+1}^{r_2+1} Y_{l,i} \right) + \sum_{(r_1=r_2)\text{is an } l\text{-epoch}} Y_{l,r_2+1} =$$

$$\sum_{(r_1<r_2)\text{is an } l\text{-epoch}} \left(\sum_{i=r_1+1}^{r_2} Y_{l,i} \right) + \sum_{(r_1,r_2)\text{is an } l\text{-epoch}} Y_{l,r_2+1} \leq$$

$$\frac{f_l}{\lambda} \sum_{(r_1<r_2)\text{is an } l\text{-epoch}} \left(\sum_{i=r_1}^{r_2-1} f_{l,i} \right) + \sum_{(r_1,r_2)\text{is an } l\text{-epoch}} f_{l,r_2+1} \leq$$

$$\frac{f_l^2}{\lambda} + f_l.$$

Lemma 6. $P(Y_l > 0) \leq \frac{f_l}{t}$.

Proof. Since I_i are independent and $0 \leq \frac{f_{l,i}}{t} \leq 1$ we can apply Fact 8:

$$P(\cap_{i=1}^{r}(m_{i,I_i} \neq l)) = \prod_{i=1}^{r}(1 - \frac{f_{l,i}}{t}) \geq (1 - \frac{f_l}{t}).$$

Thus,

$$P(Y_l > 0) \leq P(\cup_{i=1}^{r}(m_{i,I_i} = l)) \leq \frac{f_l}{t}. \tag{25}$$

Lemma 7. *There exists an absolute constant α such that (13) implies that $P(A_{i,j}) \leq 0.01$ for at least $0.99f_1$ pairs $(i,j) \in Q$.*

Proof. From Lemmas 5, 6:

$$E(Y_l) \leq \frac{f_l}{t}(\frac{f_l^2}{\lambda} + f_l),$$

$$E(Y) = \sum_{l=2}^{n} E(Y_l) \leq \frac{G_3}{\lambda t} + \frac{G_2}{t}.$$

If follows that $\sum_{(i,j)\in Q} \mathbf{1}_{A_{i,j}} = Y$. Recall that by (13):

$$|Q| = f_1 \geq \alpha(\frac{G_3}{\lambda t} + \frac{G_2}{t}) \geq \alpha E(\sum_{(i,j)\in Q} \mathbf{1}_{A_{i,j}}).$$

To summarize Y is the sum of f_1 indicators such that $E(Y) \leq 10^{-4}f_1$ for $\alpha \geq 10^4$. We apply Fact 9 with $u = f_1, \mu = 10^4$ and obtain that there must exist at least $0.99f_1$ indicators with expected value at most 0.01. The lemma follows.

The following fact is a well known. For completeness we present the proof.

Fact 8 *Let $\alpha_1, \ldots, \alpha_r$ be real numbers in $[0,1]$. Then*

$$\prod_{i=1}^{r}(1 - \alpha_i) \geq 1 - (\sum_{i=1}^{r}\alpha_i).$$

Proof. If $\sum_{i=1}^{r}\alpha_i \geq 1$ then

$$\prod_{i=1}^{r}(1 - \alpha_i) \geq 0 \geq 1 - (\sum_{i=1}^{r}\alpha_i).$$

Thus we can assume that $\sum_{i=1}^{r}\alpha_i < 1$. We will prove the claim by induction on r. For $r = 2$ we obtain $(1-\alpha_1)(1-\alpha_2) = (1 - \alpha_1 - \alpha_2 x + \alpha_1\alpha_2) \geq (1 - \alpha_1 - \alpha_2)$. For $r > 2$, we have, by induction,

$$\prod_{i=1}^{r}(1 - \alpha_i) \geq (1 - (\sum_{i=1}^{r-1}\alpha_i))(1 - \alpha_r) \geq 1 - (\sum_{i=1}^{r}\alpha_i).$$

Fact 9 *Let $X_1, \ldots, X_u$ be a sequence of indicator random variables. Let $S = \{i : P(X_i = 1) \leq \nu\}$. If $E(\sum_{i=1}^{u} X_i) \leq \mu u$ then $|S| \geq (1 - \frac{\mu}{\nu})u$. Specifically, if $\mu \leq 10^{-4}$ then there exists $S \subseteq [u]$ such that $|S| \geq 0.99u$ and $P(X_i = 1) \leq 0.01$ for all $i \in S$.*

Proof. Indeed,

$$\mu u \geq \sum_{i \notin S} P(X_i = 1) \geq \nu(u - |S|).$$

Events of type $\boldsymbol{B}$ For $(i,j) \in Q$ let $Z_{(i,j)} = \mathbf{1}_{B_{i,j}}$. Let $Z = \sum_{(i,j)\in Q} Z_{(i,j)}$. We use arguments that are similar to the ones from the previous section. To stress the similarity we abuse the notation and denote by $Y_{l,h,(i,j)}$ the indicator of the event that $h > i + 1$, $s_h = l$ and

$$\left(d_{i,j} + \sum_{u=i+1}^{h-1} f_{1,u} \right) < c_h.$$

Define $Y_{l,h} = \sum_{(i,j)\in Q} Y_{l,h,(i,j)}$, $Y_l = \sum_{h=1}^{r} Y_{l,h}$.

Fact 10 $Y_l \leq f_l$.

Proof. Repeating the arguments from Fact 3 we have $c_h \mathbf{1}_{s_h=l} \leq f_{l,h}$ and thus $Y_{l,h} \leq f_{l,h}$.

Fact 11 $P(Y_l > 0) \leq \frac{f_l}{t}$.

Proof. The proof is identical to the proof of Lemma 6.

Lemma 12. *There exist absolute constants α, β such that (13) implies that $P(B_{i,j}) \leq 0.01$ for at least $0.99 f_1$ pairs $(i,j) \in Q$.*

Proof. Denote $Y = \sum_{l=1}^{n} Y_l$. If follows that $Z \leq Y$ and $E(Z) \leq E(Y)$. By Facts 11 and 10 if follows that $E(Y_l) \leq \frac{f_l^2}{t}$. Thus by (13):

$$E(Z) \leq E(Y) \leq \frac{F_2}{t} = \frac{G_2}{t} + f_1 \frac{f_1}{t} \leq (\alpha^{-1} + \beta) f_1.$$

Recall that we define $\alpha^{-1} + \beta \leq 10^{-4}$. Thus, we can repeat the arguments from Lemma 7.

Events of type $\boldsymbol{H}$

Definition 2. *Let $U = \{u_1, \ldots, u_t\}$ and $W = \{w_1, \ldots, w_t\}$ be two sequences of non-negative integers. Let (i,j) be a pair such that $1 \leq i \leq t$ and $1 \leq j \leq u_i$. Denote (i,j) as a* loosing *pair (w.r.t. sequences U, W) if there exists $h, i \leq h \leq t$ such that:*

$$-j + \sum_{s=i}^{h} (u_s - w_s) < 0.$$

Denote any pair that is not a loosing pair as a a winning *pair.*

In this section we consider the following pair (U, W) of sequences. For $i = 1, \ldots, r$ let $u_i = f_{1,i}$ and $w_i = \lambda$.

Fact 13 *If (i, j) is a winning pair w.r.t. (U, W) then $H_{i,j'}$ does not occur where j' is such that $m_{i,j'} = 1$ and $d_{i,j'} = f_{1,i} - j + 1$.*

Proof. By Definition 2, for every $i \le h \le r$:

$$-j + \sum_{l=i}^{h} u_l \ge \sum_{l=i}^{h} w_l. \tag{26}$$

Since $\sum_{l=i}^{h} w_i = (h - i + 1)\lambda$ and $d_{i,j'} = f_{1,i} - j + 1$ we have for every $i \le h \le r$:

$$d_{i,j'} + \sum_{l=i+1}^{h} d_{l,1} = f_{i,1} - j + 1 + \sum_{l=i+1}^{h} f_{l,1} =$$

$$-j + 1 + \sum_{l=i}^{h} u_l \ge -j + \sum_{l=i}^{h} u_l \ge \sum_{l=i}^{h} w_l = (h - i + 1)\lambda.$$

Substitute h by $h - 1$ (for $h > i$):

$$d_{i,j'} + \sum_{l=i+1}^{h-1} d_{l,1} \ge (h - i)\lambda.$$

Thus $H_{i,j'}$ does not occur, by (18).

Lemma 14. *There exists an absolute constant α such that (13) implies that $H_{i,j}$ does not occur for at least $0.99 f_1$ pairs $(i, j) \in W$.*

Proof. By Lemma 15 there exist at least

$$\sum_{i=1}^{r} (u_i - w_i)$$

winning pairs (i, j) w.r.t. the (U, W). Also, $\sum_{i=1}^{r} u_i = \sum_{i=1}^{r} f_{1,i} = f_1$ and $\sum_{i=1}^{r} w_i = \lambda r$. Thus there exist at least $f_1 - \lambda r$ winning pairs (i, j) w.r.t. the (U, W). In the statement of Fact 13 the mapping from j to j' is a bijection; thus there exist at least $f_1 - \lambda r$ pairs (i, j') s.t. $m_{i,j'} = 1$ and $H_{i,j'}$ does not occur. By (13) we have $f_1 \ge \alpha \lambda r$ and the lemma follows.

Due to the lack of space we omit the proof of the following lemma and refer the reader to [11].

Lemma 15. *If $\sum_{s=1}^{t} (u_s - w_s) > 0$ then there exist at least $\sum_{s=1}^{t} (u_s - w_s)$ winning pairs.*

3 Streaming Algorithm for Heavy Elements

Let D be a stream. Define

$$\psi = \frac{n^{1-(1/k)}G_k^{1/k}}{F_1}, \delta = 2^{\lceil 0.5\log_2(\psi)\rceil}, t = \left\lceil \frac{\delta F_1}{n^{1/k}} \right\rceil, \lambda = \left\lceil \frac{F_1\delta^3}{n} \right\rceil, \quad (27)$$

where we use (2) to define F_k. We will make the following assumptions:

$$f_1 \leq 0.1F_1, \quad t \leq F_1, \quad F_1(mod t) = 0. \quad (28)$$

Then it is possible to define a matrix a $r \times t$ matrix M, where $r = F_1/t$ and with entries $m_{i,j} = p_{it+j}$.

Theorem 2. *Let M be a $r \times t$ matrix such that (27) is true. Then there exist absolute constants*[4] *$\alpha \geq 2 * 10^4, \beta \leq 0.5 * 10^{-4}$ such that*

$$\alpha G_k^{1/k} \leq f_1 \leq \beta t \quad (29)$$

imply

$$P(S_r = 1) \geq \frac{\delta}{2n^{1-(2/k)}}. \quad (30)$$

Proof. By (29) and Facts 19, 18, 17:

$$6\alpha(\lambda r + \frac{G_3}{\lambda t} + \frac{G_2}{t}) \leq f_1 \leq \beta t.$$

Also, (27) implies $f_1/t \geq \frac{\delta}{n^{1-(2/k)}}$. Thus, (30) follows from Theorem 1.

Algorithm 1 describes our implementation of the pick-and-drop sampling.

Theorem 3. *Denote $f_i^k > 100\sum_{j\neq i} f_j^k$ as a* heavy *element. There exist a (constructive) algorithm that makes one pass over the stream and uses $O(n^{1-2/k}\log(n))$ bits. The algorithm outputs a pair $(i, \tilde{f}_i)$ such that $\tilde{f}_i \leq f_i$ with probability* 1. *If there exists a heavy element f_i then also with constant probability the algorithm will output $(i, \tilde{f}_i)$ such that $(1-\epsilon)f_i \leq \tilde{f}_i$.*

Proof. Define t as in (27). W.l.o.g., we can assume that F_1 is divisible by t. Note that if $t > F_1$ or $f_1 \geq 0.1F_1$ then it is possible to find a heavy element with $O(n^{1-2/k})$ bits by existing methods such as [14]. Otherwise, a stream D defines a matrix M for which we compute $O(n^{1-2/k}/\epsilon\delta)$ independent pick-and-drop samples. Since we do not know the value of δ we should repeat the experiment for all possible values of δ. Output the element with the maximum frequency. With constant probability the output of the pick-and-drop sampling will include a $(1-\epsilon)$ approximation of the frequency f_i. Thus, there will be no other f_j that can give a larger approximation and replace a heavy element. The total space will define geometric series that sums to $O(n^{1-2/k}\log(n))$.

If we know F_1 ahead of time then we can compute the value of t for any possible δ and thus solve the problem in one pass. However, one can show that the well-known doubling technique (when we double our parameter t each time the size of the stream doubles) will work in our case and thus one pass is sufficient even without knowing F_1.

[4] We did not try to optimize the constants.

Algorithm 1. P&D(M, r, t, λ)

Generate i.i.d. r.v. $\{I_j\}_{j=1}^{r}$ with uniform distribution on $[t]$.
$S_1 = m_{1,I_1}$,
$C_1 = d_{1,I_1}$,
$q_1 = 1$.
for $i = 2 \to r$ **do**
  compute $s_i = m_{i,I_i}, c_i = d_{i,I_i}$
  if $(C_{i-1} < \max\{\lambda q_{i-1}, c_i\})$ **then**
    $S_i = s_i$,
    $C_i = c_i$,
    $q_i = 1$
  else
    $S_i = S_{i-1}$,
    $C_i = C_{i-1} + f_{S_i,l}$,
    $q_i = q_{i-1} + 1$
  end if
end for
Output (S_r, C_r).

Recall that in [8] we develop a method of recursive sketches with the following property: given an algorithm that finds a heavy element and uses memory $\mu(n)$ (where $\mu = \Omega(n^\alpha)$ for some constant α)8, it is possible to solve the frequency moment problem in space $O(\mu(n)\log^{(c)}(n))$. In [8] we applied recursive sketches with the method of Charikar et.al. [14]. Thus, we can replace the method from [14] with Theorem 3 and obtain:

Theorem 4. *Let ϵ and k be constants. There exists a (constructive) algorithm that computes $(1 \pm \epsilon)$-approximation of F_k, uses $O(n^{1-2/k}\log(n)\log^{(c)}(n))$ memory bits, makes one pass and errs with probability at most $1/3$.*

3.1 Useful Facts and Inequalities

Fact 16 $1 \le \delta \le 2n^{(k-1)/2k}$.

Proof. Indeed, $G_1 \le G_k^{1/k} n^{1-1/k}$ by Hölder inequality and since $f_1 \le 0.1F_1$ by (28) we have $\psi \ge 0.5$; thus, $\lceil 0.5\log_2(\psi)\rceil \ge 0$ and the lower bound follows. Also, $F_k^{1/k}$ is the L_k norm for the frequency vector since since all frequencies are non-negative. Since $L_k \le L_1$ we conclude that $\psi \le n^{1-1/k}$ and the fact follows.

Observe that there exists a frequency vector with $\delta = O(1)$: put $f_j = 1$ for all $i \in [n]$. At the same time there exists a vector with $\delta = \Omega(n^{(k-1)/2k})$: put $f_1 = n$ and $f_j = 1$ for $j > 2$. It is not hard to see that if δ is sufficiently large then a naïve sampling method will find a heavy element. For example, in the latter case, the heavy element occupies half of the stream.

Fact 17 $\lambda r \le 4G_k^{1/k}$.

Proof. Recall that $F_1 = rt$. The fact follows from the definitions of λ and t.

Fact 18

$$\frac{G_2}{t} \leq G_k^{1/k}.$$

Proof. Define $\alpha = \frac{k-3}{2(k-2)}$. We have by Hölder inequality:

$$G_2^{\alpha} \leq G_k^{\frac{2\alpha}{k}} n^{\alpha(1-\frac{2}{k})} = G_k^{\frac{k-3}{k(k-2)}} n^{\frac{k-3}{2k}}. \tag{31}$$

Also, by Fact 20

$$G_2^{1-\alpha} = G_2^{\frac{k-1}{2(k-2)}} \leq G_k^{\frac{1}{2(k-2)}} G_1^{\frac{1}{2}}. \tag{32}$$

Thus,

$$G_2 \leq G_k^{\frac{k-3}{k(k-2)}} n^{\frac{k-3}{2k}} G_k^{\frac{1}{2(k-2)}} F_1^{\frac{1}{2}} =$$

$$G_k^{\frac{1}{k}} \frac{F_1}{n^{1/k}} \left(\frac{G_k^{\frac{1}{k}} n^{\frac{k-1}{k}}}{F_1} \right)^{1/2} = tG_k^{\frac{1}{k}}.$$

Fact 19 $\frac{G_3}{\lambda t} \leq G_k^{1/k}$.

Proof. By Hölder inequality,

$$G_3 \leq G_k^{3/k} n^{1-(3/k)}. \tag{33}$$

Thus

$$\frac{G_3}{\lambda t} = \frac{n^{1+(1/k)} G_3}{F_1^2 \delta^4} \leq \frac{n^{2-(2/k)} G_k^{3/k}}{F_1^2 \delta^4} \leq G_k^{1/k}.$$

Fact 20 *Let $v_1, \ldots, v_n$ be a sequence of non-negative numbers and let $k > 2$. Then*

$$\left(\sum_{i=1}^{n} v_i^2 \right)^{(k-1)} \leq \left(\sum_{i=1}^{n} v_i^k \right) \left(\sum_{i=1}^{n} v_i \right)^{(k-2)}$$

Proof. Define $\lambda_i = \frac{v_i}{\sum_{j=1}^{n} v_j}$. Since $g(x) = x^{k-1}$ is convex on the interval $[0, \infty)$ we can apply Jensen's inequality and obtain:

$$\left(\frac{\sum_{i=1}^{n} v_i^2}{\sum_{i=1}^{n} v_i} \right)^{(k-1)} = (\sum_{i=1}^{n} \lambda_i v_i)^{(k-1)} \leq (\sum_{i=1}^{n} \lambda_i v_i^{(k-1)}) = \frac{\sum_{i=1}^{n} v_i^k}{\sum_{i=1}^{n} v_i}.$$

References

1. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. J. Comput. Syst. Sci. 58(1), 137–147 (1999)
2. Andoni, A., Krauthgamer, R., Onak, K.: Streaming algorithms via precision sampling. In: FOCS, pp. 363–372 (2011)

3. Andoni, A., Nguy˜ên, H.L., Polyanskiy, Y., Wu, Y.: Tight lower bound for linear sketches of moments. In: Smotrovs, J., Yakaryilmaz, A. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 25–32. Springer, Heidelberg (2013)
4. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: An information statistics approach to data stream and communication complexity. J. Comput. Syst. Sci. 68(4), 702–732 (2004)
5. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D., Trevisan, L.: Counting distinct elements in a data stream. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 1–10. Springer, Heidelberg (2002)
6. Beame, P., Jayram, T.S., Rudra, A.: Lower bounds for randomized read/write stream algorithms. In: STOC 2007: Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing, pp. 689–698. ACM, New York (2007)
7. Braverman, V., Gelles, R., Ostrovsky, R.: How to catch l_2-heavy-hitters on sliding windows. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 638–650. Springer, Heidelberg (2013)
8. Braverman, V., Ostrovsky, R.: Generalizing the layering method of indyk and woodruff: Recursive sketches for frequency-based vectors on streams. In: Raghavendra, P., Raskhodnikova, S., Jansen, K., Rolim, J.D.P. (eds.) APPROX/RANDOM 2013, LNCS, vol. 8096, pp. 58–70. Springer, Heidelberg (2013)
9. Braverman, V., Ostrovsky, R.: Smooth histograms for sliding windows. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007, pp. 283–293. IEEE Computer Society, Washington, DC (2007)
10. Braverman, V., Ostrovsky, R.: Recursive sketching for frequency moments. CoRR, abs/1011.2571 (2010)
11. Braverman, V., Ostrovsky, R.: Approximating large frequency moments with pick-and-drop sampling. CoRR, abs/1212.0202 (2012)
12. Chakrabarti, A., Cormode, G., McGregor, A.: Robust lower bounds for communication and stream computation. In: STOC 2008: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 641–650. ACM, New York (2008)
13. Chakrabarti, A., Khot, S., Sun, X.: Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In: IEEE Conference on Computational Complexity, pp. 107–117 (2003)
14. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 693–703. Springer, Heidelberg (2002)
15. Clarkson, K.L., Woodruff, D.P.: Numerical linear algebra in the streaming model. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 205–214. ACM, New York (2009)
16. Coppersmith, D., Kumar, R.: An improved data stream algorithm for frequency moments. In: SODA, pp. 151–156 (2004)
17. Cormode, G., Datar, M., Indyk, P., Muthukrishnan, S.: Comparing data streams using hamming norms (how to zero in). IEEE Trans. on Knowl. and Data Eng. 15(3), 529–540 (2003)
18. Feigenbaum, J., Kannan, S., Strauss, M., Viswanathan, M.: An approximate l1-difference algorithm for massive data streams. In: FOCS 1999: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, p. 501. IEEE Computer Society, Washington, DC (1999)
19. Flajolet, P., Nigel Martin, G.: Probabilistic counting algorithms for data base applications. J. Comput. Syst. Sci. 31(2), 182–209 (1985)
20. Ganguly, S.: Estimating frequency moments of data streams using random linear combinations. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) APPROX and RANDOM 2004. LNCS, vol. 3122, pp. 369–380. Springer, Heidelberg (2004)

21. Ganguly, S.: Polynomial estimators for high frequency moments. CoRR, abs/1104.4552 (2011)
22. Ganguly, S.: A lower bound for estimating high moments of a data stream. CoRR, abs/1201.0253 (2012)
23. Ganguly, S., Cormode, G.: On estimating frequency moments of data streams. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) APPROX and RANDOM 2007. LNCS, vol. 4627, pp. 479–493. Springer, Heidelberg (2007)
24. Indyk, P.: Stable distributions, pseudorandom generators, embeddings, and data stream computation. J. ACM 53(3), 307–323 (2006)
25. Indyk, P., Woodruff, D.: Optimal approximations of the frequency moments of data streams. In: STOC 2005: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing, pp. 202–208. ACM, New York (2005)
26. Jayram, T.S., McGregor, A., Muthukrishnan, S., Vee, E.: Estimating statistical aggregates on probabilistic data streams. In: PODS 2007: Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 243–252. ACM, New York (2007)
27. Jayram, T.S., Woodruff, D.: Optimal bounds for johnson-lindenstrauss transforms and streaming problems with sub-constant error. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, pp. 1–10. SIAM (2011)
28. Kane, D.M., Nelson, J., Woodruff, D.P.: On the exact space complexity of sketching and streaming small norms. In: Proceedings of the 2st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010 (2010)
29. Kane, D.M., Nelson, J., Woodruff, D.P.: An optimal algorithm for the distinct elements problem. In: PODS 2010: Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems of Data, pp. 41–52. ACM, New York (2010)
30. Li, P.: Compressed counting. In: SODA 2009: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 412–421. Society for Industrial and Applied Mathematics, Philadelphia (2009)
31. Muthukrishnan, S.: Data streams: algorithms and applications. Found. Trends Theor. Comput. Sci. 1(2), 117–236 (2005)
32. Nelson, J., Woodruff, D.P.: Fast manhattan sketches in data streams. In: PODS 2010: Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems of Data, pp. 99–110. ACM, New York (2010)
33. Pick, A.D., Frankel, G.W.: A developmental study of strategies of visual selectivity. Child Development 45(4), 1162–1165 (1974)
34. Woodruff, D.P.: Optimal space lower bounds for all frequency moments. In: SODA 2004: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 167–175 (2004)
35. Woodruff, D.P.: Frequency moments. In: Encyclopedia of Database Systems, pp. 1169–1170 (2009)
36. Woodruff, D.P., Zhang, Q.: Tight bounds for distributed functional monitoring. In: Proceedings of the 44th Symposium on Theory of Computing, STOC 2012, pp. 941–960. ACM, New York (2012)

Generalizing the Layering Method of Indyk and Woodruff: Recursive Sketches for Frequency-Based Vectors on Streams

Vladimir Braverman[1,⋆] and Rafail Ostrovsky[2,⋆⋆]

[1] Johns Hopkins University, Department of Computer Science
vova@cs.jhu.edu
[2] University of California Los Angeles,
Department of Computer Science and Department of Mathematics
rafail@cs.ucla.edu

Abstract. In their ground-breaking paper, Indyk and Woodruff (STOC 05) showed how to compute the k-th frequency moment F_k (for $k > 2$) in space $O(\textit{poly-log}(n,m) \cdot n^{1-\frac{2}{k}})$, giving the first optimal result up to poly-logarithmic factors in n and m (here m is the length of the stream and n is the size of the domain.) The method of Indyk and Woodruff reduces the problem of F_k to the problem of computing heavy hitters in the streaming manner. Their reduction only requires polylogarithmic overhead in term of the space complexity and is based on the fundamental idea of "layering." Since 2005 the method of Indyk and Woodruff has been used in numerous applications and has become a standard tool for streaming computations.

We propose a new recursive sketch that generalizes and improves the reduction of Indyk and Woodruff. Our method works for any non-negative frequency-based function in several models, including the insertion-only model, the turnstile model and the sliding window model. For frequency-based functions with sublinear polynomial space complexity our reduction only requires $\log^{(c)}(n)$ overhead, where $\log^{(c)}(n)$ is the iterative log function. Thus, we improve the reduction of Indyk and Woodruff by polylogarithmic factor. We illustrate the generality of our method by several applications: frequency moments, frequency based functions, spatial data streams and measuring independence of data sets.

Keywords: Data streams, frequencies, recursion, sketches.

⋆ This work was supported in part by DARPA grant N660001-1-2-4014. Its contents are solely the responsibility of the author and do not represent the official view of DARPA or the Department of Defense.

⋆⋆ Research supported in part by NSF grants CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 58–70, 2013.

1 Introduction

The celebrated paper of Alon, Matias and Szegedy [1] defined the following *streaming* model:

Definition 1. *Let m, n be positive integers. A* stream $D = D(n, m)$ *is a sequence of size m of integers $p_1, \ldots, p_m$, where $p_i \in \{1, \ldots, n\}$. A* frequency vector *is a vector of dimensionality n with* non-negative *entries $f_i, i \in [n]$ defined as:*

$$f_i = |\{j : 1 \leq j \leq m, p_j = i\}|.$$

Definition 2. *A k-th* frequency moment *of D is defined by $F_k(D) = \sum_{i \in [n]} f_i^k$. Also $F_\infty = \max_{i \in [n]} f_i$.*

Alon, Matias and Szegedy [1] initiated the study of approximating frequency moments with sublinear memory. Their surprising and fundamental results imply that for $k \leq 2$ it is possible to approximate F_k with polylogarithmic space; and that polynomial space is necessary for $k > 2$. Today, research on frequency moments is one of the central directions for streaming; many important discoveries have been made since [1]. The incomplete list of relevant work includes [22,19,3,14,4,16,17,18,20,21,27,25,7,9,26,28,30,5,13,24].

Indyk and Woodruff in their ground-breaking paper [23] gave the first optimal, up to polylogarithmic factor, algorithm for F_k. Their presented a two-pass algorithm with space complexity of $O\left(\frac{1}{\epsilon^{12}} \cdot (\log^2 n)(\log^6 m) \cdot n^{1-\frac{2}{k}}\right)$ and then shown how their two-pass algorithm can be converted to one-pass algorithm with additional poly-log multiplicative factors. Let us describe, very informally, the fundamental approach of Indyk and Woodruff [23]. They split the frequency vector into "layers," where each layer contains all entries with frequencies between, e.g., γ^i and γ^{i+1} for a carefully chosen $\gamma > 1$. Then they approximate the contribution of each layer by sampling the stream and by finding the heavy elements that contribute to the layer. Their elegant analysis shows that such a procedure ensures a good approximation with high probability. The method of Indyk and Woodruff reduces the problem of F_k to the problem of computing heavy hitters in the streaming manner. Their reduction requires polylogarithmic overhead in term of the space complexity. Since 2005 the method of Indyk and Woodruff has been used in numerous applications and has become a standard tool for streaming computations.

Our method has been inspired by the algorithm of Indyk and Woodruff. Specifically, we ask:

Question 1. For which functions $w : R \mapsto R$, it is possible to reduce the problem of computing $\sum_{i=1}^n w(f_i)$ to the problem of finding all $j \in [n]$ such that $w(j) = \Omega(\sum_{i=1}^n w(f_i))$? More generally, given an implicit vector V, when it is possible to reduce the problem of approximating $|V|$ (the L_1 norm) to the problem of finding heavy elements?

In this paper we answer Question 1 for all non-negative functions w. For streaming applications, recursion can be helpful if it is possible to reduce computations to a *single*

instance of a smaller problem. We show that it is possible to reduce such a problem on a vector of size n to a single computation of a *random* vector of size approximately $\frac{1}{2}n$. In particular, $O(\log(n))$ overhead is sufficient for any frequency-based function. Our main technical result is shown in Theorem 1. For frequency-based functions that require polynomial space our reduction only requires $\log^{(c)}(n)$ overhead. This result is shown in Theorem 4. We illustrate the generality of our method by several applications: frequency moments, frequency based functions, spatial data streams and measuring independence of data sets.

The correctness of the basic step in our algorithm follows from elementary analysis. We then employ the basic step recursively and show that $\log(n)$ recursive calls can give an algorithm that reduces the problem of approximating the sum to the problem of finding heavy hitters. Further, it is possible to reduce the number of recursive calls $\log(n)$ to $\log\log(n)$ by applying the same argument, but stopping after $O(\log\log(n))$ steps. At the depth $O(\log\log(n))$ of the recursion, the number of positive frequencies in a corresponding vector is polylogarithmically smaller than n, with constant probability. Thus, any algorithm that works in $polylog(n,m)n^{\alpha}$ space (where $0 < \alpha < 1$ is a constant) will approximate such a vector with negligible cost. Employing such an algorithm at the bottom of $\log\log(n)$ recursion reduces the $\log(n)$ factor to a $poly(\log\log(n))$ factor. Further, the same idea may be repeated at least constant number of times; this is how we achieve our final bound. The simplest variant of the argument requires only pairwise independence, giving an algorithm that requires only 4-wise independence.

1.1 Roadmap

In Section 2 we introduce the basic argument and extend it to a special case, stuitable for streaming applications, case in Section 3. In Section 4 we describe a generic algorithm for recursive computations. In Section 5 we discuss our result and demonstrate its generality by explaining several applications.

2 Recursive Sketches

In this paper we denote by $|V|$ the L_1 norm of V, i.e., $|V| = \sum_{j\in[n]} v_j$.

Definition 3. *Heavy elements*
*Let V be a vector of dimensionality n with non-negative entries $v_i \geq 0$. Let $0 < \alpha \leq 1$. An element v_i is a α-*heavy *with respect to V if: $v_i \geq \alpha|V|$. A set $S \subseteq [n]$ is a α-*core *w.r.t. V if $i \in S$ for any α-heavy v_i.*

Lemma 1. *Let $V \in R^{[n]}$ be a fixed vector and let S be an α-core w.r.t. V. Let H be a random vector with uniform zero-one entries $h_i, i \in [n]$ that are pairwise-independent. Define*

$$X = \sum_{i\in S} v_i + 2\sum_{i\notin S} h_i v_i.$$

Then $P(|X - |V|| \geq \epsilon|V|) \leq \frac{\alpha}{\epsilon^2}$.

Proof. Clearly, $E(X) = |V|$. By the properties of variance, by pairwise independence of h_i and by the definition of α-core:

$$Var(X) = 4\sum_{i \notin S} v_i^2 Var(h_i) = \sum_{i \notin S} v_i^2 \leq \alpha |V|^2.$$

Thus, by Chebyshev inequality:

$$P(|X - |V|| \geq \epsilon |V|) \leq \frac{\alpha}{\epsilon^2}.$$

Corollary 1. *Let $V \in R^{[n]}$ be a random vector and let S be an α-core w.r.t. V. Let H be a random vector independent of V and S with uniform zero-one entries $h_i, i \in [n]$ that are pairwise-independent. Define*

$$X = \sum_{i \in S} v_i + 2\sum_{i \notin S} h_i v_i.$$

Then

$$P(|X - |V|| \geq \epsilon |V|) \leq \frac{\alpha}{\epsilon^2}.$$

Proof. For any fixed V and S the main claim is true since H is independent of V and S and by Lemma 1. Thus, the corollary follows.

Recursive Computations. Let ϕ be a parameter. Let $H_1, \ldots, H_\phi$ be i.i.d. random vectors with zero-one entries that are uniformly distributed and pairwise independent. For two vectors of dimensionality n define $Had(V, U)$ to be their Hadamard product; i.e., $Had(V, U)$ is a vector of dimensionality n with entries $v_i u_i$. Define:

$$V_0 = V, \text{ and } V_j = Had(V_{j-1}, H_j) \text{ for } j = 1, \ldots, \phi.$$

Denote by v_i^j and h_i^j the i-th entry of V_j and H_j respectfully. Let $S_0, \ldots, S_\phi$ be a sequence of subsets of $[n]$ such that S_j is an α-core of V_j. Define the sequence

$$X_j = \sum_{i \in S_j} v_i^j + 2\sum_{i \notin S_j} h_i^{j+1} v_i^j, \qquad j = 0, \ldots, \phi - 1,$$

and $X_\phi = |V_\phi|$.

Fact 2

$$P(\bigcup_{j=0}^{\phi} (|X_j - |V_j|| \geq \epsilon |V_j|)) \leq \frac{(\phi + 1)\alpha}{\epsilon^2}.$$

Proof. Consider fixed $j < k$. It follows from the definitions that H_{j+1} is independent of V_j and S_j. Applying Corollary 1 and the union bound we obtain the proof.

Consider the following recursive definition:

$$Y_\phi = X_\phi, \quad Y_j = 2Y_{j+1} + \sum_{i \in S_j} (1 - 2h_i^{j+1}) v_i^j.$$

Lemma 3. *For any ϕ, γ, vector V and $\alpha = \Omega(\frac{\gamma^2}{\phi^3})$:*

$$P(|Y_0 - |V|| \geq \gamma |V|) \leq 0.2.$$

Proof. Denote $Err_j^1 = |V_j| - X_j$ and $Err_j^2 = |V_j| - Y_j$. We can rewrite

$$X_j = 2|V_{j+1}| + \sum_{i \in S_j} (1 - 2h_i^{j+1}) v_i^j.$$

Thus $X_j - Y_j = 2(|V_{j+1}| - Y_{j+1}) = 2Err_{j+1}^2$ and

$$|Err_j^2| = |Y_j - |V_j|| \leq |X_j - |V_j|| + |X_j - Y_j| = |Err_j^1| + 2|Err_{j+1}^2|.$$

By definition $Err_\phi^1 = Err_\phi^2 = 0$. Thus we can rewrite:

$$|Err_0^2| \leq |Err_0^1| + 2|Err_1^2| \leq \cdots \leq \sum_{j=0}^{\phi} 2^j |Err_j^1|.$$

Choose $\epsilon = \frac{\gamma}{10(\phi+1)}$; we have by Fact 2:

$$P(|Y_0 - |V|| \geq \gamma |V|) = P(|Err_0^2| \geq \gamma |V|) \leq P(\sum_{j=0}^{\phi} 2^j |Err_j^1| \geq \gamma |V|) \leq$$

$$P\left(\left(\sum_{j=0}^{\phi} 2^j |Err_j^1| \geq \gamma |V|\right) \cap \left(\bigcap_{j=0}^{\phi} \left(|Err_j^1| < \epsilon |V_j|\right)\right)\right) + P\left(\bigcup_{j=0}^{\phi} \left(|X_j - |V_j|| \geq \epsilon |V_j|\right)\right) \leq$$

$$P\left(\sum_{j=0}^{\phi} 2^j |V_j| \geq 10(\phi+1)|V|\right) + \frac{(\phi+1)\alpha}{\epsilon^2}.$$

For $j > 0$ we note that $|V_j|$ is a random variable defined as:

$$|V_j| = \sum_{i \in [n]} v_i \left(\prod_{t=1}^{j} h_i^t\right).$$

Since all H_j are mutually independent, we conclude that

$$E(\sum_{j=0}^{\phi} 2^j |V_j|) = \sum_{j=0}^{\phi} 2^j \left(\sum_{i \in [n]} v_i \left(\prod_{t=1}^{j} E(h_i^t)\right)\right) = \sum_{j=0}^{\phi} 2^j \left(\sum_{i \in [n]} v_i 2^{-j}\right) = (\phi+1)|V|.$$

Thus, and by Markov inequality, we have

$$P(\sum_{j=0}^{\phi} 2^j |V_j| \geq 10(\phi+1)|V|) \leq 0.1.$$

Also, $\frac{(\phi+1)\alpha}{\epsilon^2} \leq 0.1$ for sufficiently large $\alpha = \Omega(\frac{\gamma^2}{\phi^3})$. Thus,

$$P(|Y_0 - |V|| \geq \gamma |V|) \leq 0.2.$$

3 An Extension: Approximate and Random Cores

There are many ways to extend our basic result. We will explore one direction, when the cores are random and contain approximations of heavy hitters with high probability[1]. We consider vectors from a finite domain $[m]^n$.

Definition 4. *Let Ω be a finite set of real numbers. Define $Pairs_t$ to be a set of all sets of pairs of the form:*

$$\{(i_1, w_1), \ldots, (i_t, w_t)\}, \quad 1 \leq i_1 < i_2 < \ldots i_t \leq n, i_j \in N, w_j \in \Omega.$$

Further define

$$Pairs = \emptyset \cup \left(\bigcup_{t=1}^{n} Pairs_t \right).$$

Definition 5. *A non-empty set $Q \in Pairs_t$, i.e., $Q = \{(i_1, w_1), \ldots, (i_t, w_t)\}$ for some $t \in [n]$, is (α, ϵ)-cover w.r.t. vector $V \in [M]^n$ if the following is true:*

1. *$\forall j \in [t](1 - \epsilon)v_{i_j} \leq w_j \leq (1 + \epsilon)v_{i_j}$.*
2. *$\forall i \in [n]$ if v_i is α-heavy then $\exists j \in [t]$ such that $i_j = i$.*

Definition 6. *Let $\mathcal{D}$ be a probability distribution on $Pairs$. Let $V \in [m]^n$ be a fixed vector. We say that $\mathcal{D}$ is δ-good w.r.t. V if for a random element Q of Pairs with distribution D the following is true:*

$$P(Q \text{ is } (\alpha, \epsilon)\text{-cover of } V) \geq 1 - \delta.$$

Definition 7. *Let g be a mapping from $[M]^n$ to a set of all distributions on $Pairs$. We say that g is δ-good if for any fixed $V \in [M]^n$ the distribution $g(V)$ is δ-good w.r.t. V. Intuitively, g represents an output of an algorithm that finds heavy hitters (and their approximations) of input vector V w.p. $1 - \delta$.*

Definition 8. *For non-empty $Q \in Pairs$ define $Ind(Q)$ to be the set of indexes of Q. Formally, for $Q \in Pairs$, denote $Ind(Q) = \{i : \exists j < t$ such that for j-th pair (i_j, w_j) of Q it is true that $i_j = i\}$. For $i \in Ind(Q)$ denote by $w_Q(i)$ the corresponding approximation, i.e. 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, denote $w_Q(i) = 0$ for $i \notin Ind(Q)$ and $Ind(\emptyset) = \emptyset$.*

Now we are ready to repeat the arguments from the previous section.

Corollary 2. *Let $V \in R^{[n]}$ be a random vector. Let g be a δ-good mapping and let Q be a random element of $Pairs$ that is chosen according to a distribution $g(V)$. Let H be a random vector independent of V and Q with uniform zero-one entries $h_i, i \in [n]$ that are pairwise-independent. Define*

$$X' = \sum_{i \in Ind(Q)} v_i + 2 \sum_{i \notin Ind(Q)} h_i v_i.$$

[1] In this section we limit our discussion to finite sets and discrete distributions. This limitation is artificial but sufficient for our applications; on the other hand it simplifies the presentation.

Then

$$P(|X' - |V|| \geq \epsilon |V|) \leq \frac{\epsilon}{\alpha^2} + \delta.$$

Proof. Consider a fixed vector V_0 and an event that $V = V_0$. Conditioned on this event, the distribution $g(V)$ is fixed and δ-good w.r.t. V_0. Consider the event that $Q = Q_0$, where Q_0 is an (α, ϵ)-cover w.r.t. V_0. Conditioned on this event, $Ind(Q)$ is an α-cover w.r.t. V_0. Since H is independent of Q the claim is true for any such V_0 by Lemma 1 and by union bound. Thus, the corollary follows.

Recursive Computations Let ϕ be a parameter. Let $H_1, \ldots, H_\phi$ be i.i.d. random vectors with zero-one entries that are uniformly distributed and pairwise independent. Define:

$$V_0 = V, \text{ and } V_j = Had(V_{j-1}, H_j) \text{ for } j = 1, \ldots, \phi.$$

Denote by v_i^j and h_i^j the i-th entry of V_j and H_j respectfully. Let g be a δ-good mapping and let Q_i be a random element of Pairs with distribution $g(V_i)$. Define $w_j(i) = w_{Q_j}(i)$. Define the sequence:

$$X'_j = \sum_{i \in Ind(Q_j)} v_i^j + 2 \sum_{i \notin Ind(Q_j)} h_i^{j+1} v_i^j, \quad j = 0, \ldots, \phi - 1,$$

and $X'_\phi = |V_\phi|$. From Corollary 2 and by repeating the arguments from Fact 2 we obtain

Fact 4

$$P(\bigcup_{j=0}^{\phi} \left(|X'_j - |V_j|| \geq \epsilon |V_j|\right)) \leq (\phi + 1)(\frac{\alpha}{\epsilon^2} + \delta).$$

Consider the following recursive definition. Let $Y'_\phi = Y'_\phi(V_\phi)$ be a random variable that depends on random vector V_ϕ and such that for any fixed V_ϕ:

$$P(|Y'_\phi - |V_\phi|| \geq \epsilon |V_\phi|) \leq \delta.$$

Also, define for $j = 0, \ldots, \phi - 1$:

$$Y'_j = 2Y'_{j+1} + \sum_{i \in Ind(Q_j)} (1 - 2h_i^{j+1}) w_i^j.$$

Lemma 5. *For any ϕ, γ, vector V; for $\alpha = \Omega(\frac{\gamma^2}{\phi^3})$ and $\delta = \Omega(\frac{1}{\phi})$:*

$$P(|Y'_0 - |V|| \geq \gamma |V|) \leq 0.2.$$

Proof. Denote $Err_j^1 = |V_j| - X'_j$, $Err_j^2 = |V_j| - Y'_j$ and $Err_j^3 = \sum_{i \in Ind(Q_j)} |w_j(i) - v_i^j|$. We can rewrite

$$X'_j = 2|V_{j+1}| + \sum_{i \in Ind(Q_j)} (1 - 2h_i^{j+1}) v_i^j.$$

Thus $|X'_j - Y'_j| \leq 2|Err^2_{j+1}| + |Err^3_j|$ and

$$|Err^2_j| = |Y'_j - |V_j|| \leq |X'_j - |V_j|| + |X'_j - Y'_j| \leq |Err^1_j| + |Err^3_j| + 2|Err^2_{j+1}|.$$

Thus we can rewrite:

$$|Err^2_0| \leq |Err^1_0| + |Err^3_0| + 2|Err^2_1| \leq \cdots \leq 2^k Err^2_\phi + \sum_{j=0}^{\phi} 2^j |Err^1_j| + \sum_{j=0}^{\phi} 2^j |Err^3_j|.$$

Choose $\epsilon = \frac{\gamma}{30(\phi+1)}$ and denote $Z = 2^k Err^2_\phi + \sum_{j=0}^{\phi} 2^j |Err^1_j| + \sum_{j=0}^{\phi} 2^j |Err^3_j|$. Then

$$P(|Y'_0 - |V|| \geq \gamma|V|) = P(|Err^2_0| \geq \gamma|V|) \leq P(Z \geq \gamma|V|) \leq$$

$$P\left((Z \geq \gamma|V|) \cap \left(\bigcap_{j=0}^{\phi} \left(|Err^1_j| < \epsilon|V_j|\right)\right) \cap \left(\bigcap_{j=0}^{\phi} \left(|Err^3_j| < \epsilon|V_j|\right)\right) \cap \left(|Err^2_\phi| < \epsilon|V_\phi|\right)\right) +$$

$$P\left(|Err^2_\phi| \geq \epsilon|V_\phi|\right) + P\left(\bigcup_{j=0}^{\phi} \left(|Err^1_j| \geq \epsilon|V_j|\right)\right) + P\left(\bigcup_{j=0}^{\phi} \left(|Err^3_j| \geq \epsilon|V_j|\right)\right).$$

Note that by the definition of Y'_ϕ, we have $P(|Err^2_\phi| \geq \epsilon|V_\phi|) \leq \delta$. Also, by the definition of Q_j and union bound,

$$P(\bigcup_{j=0}^{\phi} (|Err^3_j| \geq \epsilon|V_j|)) \leq (\phi+1)\delta.$$

Thus and by Fact 4:

$$P(|Y'_0 - |V|| \geq \gamma|V|) \leq P\left(\sum_{j=0}^{\phi} 2^j |V_j| \geq 10(\phi+1)|V|\right) + (\phi+2)(\frac{\alpha}{\epsilon^2} + 2\delta).$$

The lemma follows by repeating the concluding arguments from Lemma 3.

4 A Generic Algorithm

Let D be a stream as in Definition 1. For a function $H : [n] \mapsto \{0,1\}$, define D_H to be a sub-stream of D that contains only elements $p \in D$ such that $H(p) = 1$. Let $V = V(D)$ be an implicit vector of dimensionality n defined by a stream, e.g., a frequency moment vector from Definition 1. We say that a vector V is *separable* if for any H, we have $Had(V(D), H) = V(D_H)$. Let $HH(D, \alpha, \epsilon, \delta)$ be an algorithm that produces (α, ϵ)-cover w.r.t. $V(D)$ w.p. $1 - \delta$, i.e., produces δ-good distribution w.r.t. $V(D)$ for some suitable finite set of Pairs, as defined in Definition 4.

Algorithm 6 . *Recursive Sum[0]*(D, ϵ)

1. *Generate* $\phi = O(\log(n))$ *pairwise independent zero-one vectors* $H_1, \ldots, H_\phi$. *Denote* D_j *to be a stream* $D_{H_1 H_2 \ldots H_j}$.
2. *Compute, in parallel, random cores* $Q_j = HH(D_j, \frac{\phi^3}{\epsilon^2}, \epsilon, \frac{1}{\phi})$
3. *If* $F_0(V_\phi) > 10^{10}$ *then output* 0 *and stop. Otherwise compute precisely* $Y_\phi = |V_\phi|$.
4. *For each* $j = \phi - 1, \ldots, 0$, *compute*

$$Y_j = 2Y_{j+1} - \sum_{i \in Ind(Q_j)} (1 - 2h_i^j) w_{Q_j}(i).$$

5. *Output* Y_0.

Theorem 1. *Algorithm 6 computes* $(1 \pm \epsilon)$*-approximation of* $|V|$ *and errs w.p. at most* 0.3. *The algorithm uses* $O(\log(n)\mu(n, \frac{1}{\epsilon^2 \log^3(n)}, \epsilon, \frac{1}{\log(n)}))$ *memory bits, where* μ *is the space required by the above algorithm* HH.

Proof. The correctness follows directly from the description of the algorithm and Lemma 5 and Markov inequality. The memory bounds follows from the direct computations.

5 Discussion and Applications

We propose a new recursive sketch that generalizes and improves the reduction of Indyk and Woodruff. Our method works for any non-negative frequency-based function in several models, including the insertion-only model, the turnstile model and the sliding window model. For frequency-based functions with sublinear polynomial space complexity our reduction requires $O(\log^{(c)}(n))$ overhead. We believe that there are many other potential applications for our method, e.g., the algorithms that currently employ the method of Indyk and Woodruff. Improving the bounds for these problems is an interesting direction for the future work. Reducing the factor to $o(\log^{(c)}(n))$ is another important open question.

5.1 Approximating Large Frequency Moments on Streams with CountSketch

We apply our technique to the problem of frequency moments.

Fact 7 *Let* V *be a vector of dimensionality* n *with non-negative entries and let* n_0 *be a number of non-zero entries in* V. *Let* $0 < \alpha < 1$ *and let* v_i *be such that* $v_i^k \geq \alpha \sum_{j \in [n]} v_j^k$. *Then* $v_i^2 \geq 0.5 \alpha^{\frac{2}{k}} n_0^{\frac{2}{k}-1} \sum_{j \neq i} v_j^2$.

Proof. If $n_0 = 0$ the fact is trivial. Otherwise, by Hölder's inequality, $\sum_{j \neq i} v_j^2 \leq n_0^{1-\frac{2}{k}} \left(\sum_{j \neq i} v_j^k \right)^{\frac{2}{k}} \leq n_0^{1-\frac{2}{k}} \alpha^{-\frac{2}{k}} v_i^2$.

The famous Count-Sketch [15] algorithm finds all α-heavy elements. In particular, the following is a corollary from [15].

Theorem 2. *(from [15]) Let a_t be the frequency of the t-th most frequent element. There exists an algorithm that w.p. $1-\delta$ outputs t pairs (i, f_i') such that $(1-\epsilon)f_i \leq f_i' \leq (1+\epsilon)f_i$ and such that all elements with $f_i \geq (1-\epsilon)a_t$ appear in the list. The algorithm uses $O((t + \frac{\sum_{i\in[n], f_i<a_t} f_i^2}{(\epsilon a_t)^2})\log(m/\delta)\log(m))$ memory bits.*

Combining with Fact 7 we obtain

Corollary 3. *There exists an algorithm that w.p. $1-\delta$ outputs $O(\alpha^{-1})$ pairs $(i, f_i'^k)$ such that $(1-\epsilon)f_i^k \leq f_i'^k \leq (1+\epsilon)f_i^k$ and such that all elements with $f_i^k \geq \alpha\sum_{j\in[n]} f_j^k$ appear in the list. The algorithm uses $O((\alpha^{-1} + \frac{k^2}{\epsilon^2}\alpha^{-2/k}n^{1-2/k})\log(m/\delta)\log(m))$ memory bits.*

The algorithm from Corollary 3 defines a δ-good distribution w.r.t. to the input vector $V(D)$ over some finite set[2] from Definition 4. Denote the algorithm from Corollary 3 by $CS(D, \alpha, \epsilon, \delta)$. Thus, combining with Algorithm 6 if gives an algorithm errs w.p. δ, outputs $(1\pm\epsilon)$-approximation of F_k and uses $O(\frac{k^2}{\epsilon^{2+4/k}}n_0^{1-2/k}\log(mn)\log(m)\log^{1+6/k}(n)\log(1/\delta))$ memory bits, nearly matching the bound in [6]. Denote this algorithm by $\mathcal{A}_0(D, \epsilon, \delta)$. We can improve the bound further recursively:

Algorithm 8 . *Recursive F_k[1]*(D, ϵ)

1. *Generate $\phi = O(\log\log(n))$ pairwise independent zero-one vectors $H_1, \ldots, H_\phi$. Denote D_j to be a stream $D_{H_1H_2\ldots H_\phi}$.*
2. *Compute, in parallel, $Q_j = CS(D_j, \frac{\epsilon^2}{\phi^3}, \epsilon, \frac{1}{100\phi})$*
3. *Compute $Y_\phi = \mathcal{A}_0(D_\phi, \epsilon, 0.1)$.*
4. *For each $j = \phi-1, \ldots, 0$, compute*

$$Y_j = 2Y_{j+1} - \sum_{i\in Ind(Q_j)} (1-2h_i^j)w_{Q_j}(i).$$

5. *Output Y_0.*

There exists a constant c such that for $\phi = c\log\log(n)$, except with a small constant probability, $F_0(D_\phi) \leq \frac{n}{\log^{10}(n)}$. Thus, executing $\mathcal{A}_0$ for $n' = \frac{n}{\log^{10}(n)}$ we obtain an approximation of $F_k(D_\phi)$ using $O(\frac{k^2}{\epsilon^{2+4/k}}n^{1-2/k}\log(mn)\log(m))$ memory bits. Since $\phi = O(\log\log(n))$, the complexity of the new algorithm becomes $O(\frac{k^2}{\epsilon^{2+4/k}}n^{1-2/k}\log(m\log(n))\log(m)(\log\log(n))^4)$. Repeating this argument a constant number of times we arrive at[3]:

[2] Indeed, we can define the finite set Ω from Definition 4 as a set of all possible outputs of Count-Sketch executed over all vectors on $[m]^n$. This is a finite set (for finite n, m) and thus we can define Pairs accordingly.

[3] We note that this algorithm was proposed in the previous version of the paper [11].

Theorem 3. *Define $g_1(n) = \log(n)$ and $g_t(n) = \log(g_{t-1}(n))$. For any constant t there exist an algorithm computes a $(1 \pm \epsilon)$-approximation of $F_k(D)$, errs w.p. at most $\frac{1}{3}$ and uses $O(c_t k^2 \epsilon^{-2-(4/k)} n^{1-\frac{2}{k}} g_t(n) \log^2(m))$ memory bits, where c_t is a constant that depends on t.*

5.2 Recursive Sketches for Frequency-Based Functions with Sublinear Polynomial Space Complexity

Theorem 3 can be generalized to any non-negative frequency based functions with sublinear polynomial space complexity. In particular, we show that the problem of approximating $|V|$ and finding heavy hitters are almost equivalent.

Theorem 4. *Define $g_1(n) = \log(n)$ and $g_t(n) = \log(g_{t-1}(n))$. Let V a vector such that it is possible to find heavy elements using space $S(n) = \Omega(n^\alpha)$ for some constant α such that $0 < \alpha < 1$. Then for any constant ϵ and for any constant t there exist an algorithm that computes $(1 \pm \epsilon)$-approximation of $|V|$, errs w.p. at most $\frac{1}{3}$ and uses $O(c_t n^\alpha g_t(n))$ memory bits, where c_t is a constant that depends on t and ϵ.*

5.3 Other Models of Streaming Computations

Our method can be directly translated to any model that allows separability and preserves non-negative values of the implicit vector. Specifically, we need the non-negativity to apply the Markov inequality. For example, we can apply the reduction to the turnstile model where we observe two streams D_1 and D_2 and need to compute the $\sum_{i=1}^{n} w(|u_i - z_i|)$ where u_i and z_i are frequencies in D_1 and D_2. Similarly, our reduction will work for the sliding window model. For example, it should improve (by polylog factor) the results for the frequency moments from [9].

5.4 Approximating Large Frequency Moments on Streams with Pick-and-Drop Sampling

In [8] we combine our method of recursive sketches with the pick-and-drop sampling and compute F_k with $O(n^{1-2/k} \log(n) \log^{(c)}(n))$ bits. We reduce the ratio between the upper and lower bounds from $O(\log^2(n))$ to $O(\log(n) \log^{(c)}(n))$. Thus, we provide a (roughly) quadratic improvement of the result of Andoni, Krauthgamer and Onak [2]. To the best of our knowledge, this is the best currently known result for constant ϵ and for insertion-only streams.

5.5 Spatial Data Streams

Recursive sketching is not limited to the frequency moments or to the insertion-only streams. For example, the method has found applications in the work of Tirthapura and Woodruff [29] on spatial data streams. Specifically, Tirthapura and Woodruff say "We choose to follow [11] since it provides a simpler exposition and has several properties we will exploit."

5.6 Frequency-Based Functions and Measuring Independence of Datasets

In [12] we consider zero-one frequency laws for the frequency-based functions. As one of the key steps we employ general reduction from sums to heavy hitters. Our method follows the ideas of Indyk and Woodruff and involves large polylogarithmic factors. Replacing our method in [12] we should be able to achieve polylogarithmic improvements in space. Similar polylogarithmic improvement should be possible for measuring independence (in terms of total variation distanced) of datasets [10].

References

1. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. J. Comput. Syst. Sci. 58(1), 137–147 (1999)
2. Andoni, A., Krauthgamer, R., Onak, K.: Streaming algorithms via precision sampling. In: FOCS, pp. 363–372 (2011)
3. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: An information statistics approach to data stream and communication complexity. J. Comput. Syst. Sci. 68(4), 702–732 (2004)
4. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D., Trevisan, L.: Counting distinct elements in a data stream. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 1–10. Springer, Heidelberg (2002)
5. Beame, P., Jayram, T.S., Rudra, A.: Lower bounds for randomized read/write stream algorithms. In: STOC 2007: Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing, pp. 689–698. ACM, New York (2007)
6. Bhuvanagiri, L., Ganguly, S., Kesh, D., Saha, C.: Simpler algorithm for estimating frequency moments of data streams. In: SODA, pp. 708–713 (2006)
7. Braverman, V., Gelles, R., Ostrovsky, R.: How to catch l_2-heavy-hitters on sliding windows. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 638–650. Springer, Heidelberg (2013)
8. Braverman, V., Ostrovsky, R.: Generalizing the layering method of indyk and woodruff: Recursive sketches for frequency-based vectors on streams. In: Raghavendra, P., Raskhodnikova, S., Jansen, K., Rolim, J.D.P. (eds.) APPROX/RANDOM 2013. LNCS, vol. 8096, pp. 58–70. Springer, Heidelberg (2013)
9. Braverman, V., Ostrovsky, R.: Smooth histograms for sliding windows. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007, pp. 283–293. IEEE Computer Society, Washington, DC (2007)
10. Braverman, V., Ostrovsky, R.: Measuring independence of datasets. In: Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, pp. 271–280. ACM, New York (2010)
11. Braverman, V., Ostrovsky, R.: Recursive sketching for frequency moments. CoRR, abs/1011.2571 (2010)
12. Braverman, V., Ostrovsky, R.: Zero-one frequency laws. In: Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, pp. 281–290. ACM, New York (2010)
13. Chakrabarti, A., Cormode, G., McGregor, A.: Robust lower bounds for communication and stream computation. In: STOC 2008: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 641–650. ACM, New York (2008)
14. Chakrabarti, A., Khot, S., Sun, X.: Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In: IEEE Conference on Computational Complexity, pp. 107–117 (2003)

15. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 693–703. Springer, Heidelberg (2002)
16. Coppersmith, D., Kumar, R.: An improved data stream algorithm for frequency moments. In: SODA, pp. 151–156 (2004)
17. Cormode, G., Datar, M., Indyk, P., Muthukrishnan, S.: Comparing data streams using hamming norms (how to zero in). IEEE Trans. on Knowl. and Data Eng. 15(3), 529–540 (2003)
18. Feigenbaum, J., Kannan, S., Strauss, M., Viswanathan, M.: An approximate l1-difference algorithm for massive data streams. In: FOCS 1999: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, p. 501. IEEE Computer Society, Washington, DC (1999)
19. Flajolet, P., Nigel Martin, G.: Probabilistic counting algorithms for data base applications. J. Comput. Syst. Sci. 31(2), 182–209 (1985)
20. Ganguly, S.: Estimating frequency moments of data streams using random linear combinations. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) APPROX and RANDOM 2004. LNCS, vol. 3122, pp. 369–380. Springer, Heidelberg (2004)
21. Ganguly, S., Cormode, G.: On estimating frequency moments of data streams. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) APPROX and RANDOM 2007. LNCS, vol. 4627, pp. 479–493. Springer, Heidelberg (2007)
22. Indyk, P.: Stable distributions, pseudorandom generators, embeddings, and data stream computation. J. ACM 53(3), 307–323 (2006)
23. Indyk, P., Woodruff, D.L.P.: Optimal approximations of the frequency moments of data streams. In: STOC 2005: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing, pp. 202–208. ACM, New York (2005)
24. Jayram, T.S., McGregor, A., Muthukrishnan, S., Vee, E.: Estimating statistical aggregates on probabilistic data streams. In: PODS 2007: Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 243–252. ACM, New York (2007)
25. Kane, D.M., Nelson, J., Woodruff, D.P.: On the exact space complexity of sketching and streaming small norms. In: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010 (2010)
26. Kane, D.M., Nelson, J., Woodruff, D.P.: An optimal algorithm for the distinct elements problem. In: PODS 2010: Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems of Data, pp. 41–52. ACM, New York (2010)
27. Li, P.: Compressed counting. In: SODA 2009: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 412–421. Society for Industrial and Applied Mathematics, Philadelphia (2009)
28. Nelson, J., Woodruff, D.P.: Fast manhattan sketches in data streams. In: PODS 2010: Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems of Data, pp. 99–110. ACM, New York (2010)
29. Tirthapura, S., Woodruff, D.P.: Rectangle-efficient aggregation in spatial data streams. In: Proceedings of the 31st Symposium on Principles of Database Systems, PODS 2012, pp. 283–294. ACM, New York (2012)
30. Woodruff, D.P.: Optimal space lower bounds for all frequency moments. In: SODA 2004: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 167–175 (2004)

Capacitated Network Design on Undirected Graphs

Deeparnab Chakrabarty[1], Ravishankar Krishnaswamy[2,⋆], Shi Li[3], and Srivatsan Narayanan[4]

[1] Microsoft Research India
dechakr@microsoft.com
[2] Princeton University
ravishan@cs.cmu.edu
[3] Princeton University
shili@cs.princeton.edu
[4] Carnegie Mellon University
srivatsa@cs.cmu.edu

Abstract. In this paper, we study the approximability of *the capacitated network design problem* (Cap-NDP) on *undirected* graphs: Given $G = (V, E)$ with non-negative costs c and capacities u on its edges, source-sink pairs (s_i, t_i) with demand r_i, the goal is to find the minimum cost subgraph where the minimum (s_i, t_i) cut with u-capacities is at least r_i. When $u \equiv 1$, we get the usual SNDP for which Jain gave a 2-approximation algorithm [9]. Prior to our work, the approximability of undirected Cap-NDP was not well understood even in the single source-sink pair case. In this paper, we show that the single-source pair Cap-NDP is label-cover hard in undirected graphs.

An important special case of single source-sink pair undirected Cap-NDP is the following *source location problem*. Given an undirected graph, a collection of sources S and a sink t, find the minimum cardinality subset $S' \subseteq S$ such that $\mathsf{flow}(S', t)$, the maximum flow from S' to t, equals $\mathsf{flow}(S, t)$. In general, the problem is known to be set-cover hard. We give a $O(\rho)$-approximation when $\mathsf{flow}(s, t) \approx_\rho \mathsf{flow}(s', t)$ for $s, s' \in S$, that is, all sources have max-flow values to the sink within a multiplicative ρ factor of each other.

The main technical ingredient of our algorithmic result is the following theorem which may have other applications. Given a capacitated, undirected graph G with a dedicated sink t, call a subset $X \subseteq V$ *irreducible* if the maximum flow $f(X)$ from X to t is strictly greater than that from any strict subset $X' \subset X$, to t. We prove that for any irreducible set, X, the flow $f(X) \geq \frac{1}{2} \sum_{i \in X} f_i$, where f_i is the max-flow from i to t. That is, undirected flows are quasi-additive on irreducible sets.

1 Introduction

In the capacitated network design problem (Cap-NDP), we are given a graph $G = (V, E)$. Each edge e has a cost $c(e)$ and a capacity $u(e)$ which we assume to be non-negative integers. We are also given a collection of pairs of *terminals* $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$, where each s_i and t_i lies in V. Each pair is associated with an integer r_i. The objective is to find a *minimum cost* subgraph of G in which every s_i can send a flow

⋆ The author is supported by a Simons Postdoctoral Fellowship.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 71–80, 2013.

of at least r_i units to t_i. We are *not* requiring these flows to be satisfied concurrently; the concurrent requirement leads to a different problem.

The problem generalizes many problems, the simplest of which is probably the minimum knapsack problem induced when the graph has two nodes and parallel edges between them. When all the capacities are unit, and the graph is undirected, then the problem is what is called the *survivable* network design problem (SNDP) for which a 2-approximation is known [9]. SNDP is label-cover hard [4] in case of directed graphs; this already shows the hardness of the above problem for directed networks. In fact, Cap-NDP is label-cover hard for directed graphs even when there is a single pair of terminals [5,3].

For undirected graphs, algorithmically nothing better is known, and the hardness results were weaker. Hajiaghayi et al. [8] showed that the single pair Cap-NDP is as hard as the group Steiner tree problem. This implies a $\Omega(\log^2 n)$ hardness. At this point we should stress that we are *disallowing* picking multiple copies of an edge; if this were allowed, then a $O(\log k)$-approximation algorithm is known for undirected graphs [3]. In this paper, we prove that the single source-sink pair Cap-NDP is label-cover hard even in undirected graphs. More precisely, unless $NP \subseteq DTIME(n^{\text{polylog}n})$, for any $\delta > 0$, there is no $2^{\log^{1-\delta} n}$-approximation for the problem.

1.1 A Source Location Problem

Consider a capacitated network with a dedicated sink t and a subset of terminals $S \subseteq V$. Given a subset $X \subseteq S$, let $f(X) \in \mathbb{R}$ denote the maximum flow that can be sent from X to t. We use f_i as a shorthand for $f(\{i\})$. It is a standard fact that f is monotone and submodular: indeed, f is monotone because adding a new source can only increase the total flow, and f is submodular since the marginal flow due to a terminal decreases as X becomes larger.

We consider the following problem: find the minimum cardinality subset $X \subseteq S$ such that $f(X) = f(S)$. This is a special case of Cap-NDP where we create a super-source s_1 and connect it to each source $s \in S$ by an edge of cost 1 and capacity $f(\{s\})$. Each edge in the original graph has cost 0 and capacity equal to its original capacity. The sink t_1 is the vertex t, and the requirement is r_1 is set to $f(S)$. Then the minimum cost graph H which can support a flow of r_1 between s_1 and t_1 exactly corresponds to the minimum cardinality set of sources which can send a flow $f(S)$ to the sink t.

The above problem is a special case of a widely studied class of problems known as *source location problems*. In its generality, each vertex $v \in G$ has a cost $c(v)$ and demand $d(v)$, and the goal is to pick a minimum cost subset X such that for all v, $f(X, v) \geq d(v)$. We obtain the above problem by setting $c(v) = 1$ for $v \in S$, and infinity otherwise, and demands $d(t) = 1$, and 0 otherwise. Source location problems have been studied extensively (see, for instance, [12]), and many special cases are known to be polynomial time solvable ([11,14,15,2]). For example, Kortsarz et al. [11] showed that the problem is polynomial time solvable when the maximum demand is at at most 3, and Arata et al. showed that the problem is easy if the set S of terminals is V. In general, the problem is as hard as set cover ([13,1]), and in fact, the reductions therein show that the above special case is also set-cover hard. Algorithmically, there is a logarithmic approximation [13], and for the above problem such a result follows by noting that problem is a special case of submodular set cover [16].

Irreducible Sets. In order to understand the above problem better, in this paper we study the *behavior* of the function f on undirected graphs. In particular, given a set $X \subseteq S$, can we lower bound $f(X)$ in terms of the f_i's for $i \in X$? Our main result answers this question affirmatively for a natural class of terminal sets X which we call *irreducible sets*. A subset $X \subseteq V$ is *irreducible* if $f(X) > f(X')$ for all *strict subsets* $X' \subsetneq X$. That is, removing any vertex from X strictly decreases the maximum flow that can be sent to the sink.

We now state our main positive result which shows that on such sets, the submodular function f behaves 'almost' additively.

Theorem 1. *Given an* undirected *graph G and any irreducible set X,*

$$f(X) \geq \frac{1}{2} \sum_{i \in X} f_i.$$

Furthermore, there is an (X, t) flow of value $f(X)$ such that the total out-flow from terminal i is at least $f_i/2$ for all $i \in X$.

We note that an analogous result is not true for directed graphs; in fact, the ratio $\sum_{i \in X} f_i / f(X)$ may be as large as $|X|$. To see this consider a collection of terminals with an arc to a vertex v and an arc from vertex v to t. All these arcs have capacity 1. Furthermore, each terminal has a direct arc to t of capacity ε. Note that X is irreducible; each terminal sends nonzero flow through its 'private' arc. However, $\sum_{i \in X} f_i / f(X)$ tends to $|X|$ as ε tends to 0. It is instructive to note that if the arcs were undirected, then the set X becomes *reducible*; the direct arcs aren't private anymore and other terminals can send flow through them. Thus the above example also shows irreducibility is necessary for the above theorem to hold.

We believe the condition of irreducibility is a natural extremal condition. For instance, in the source location problem above, it is easy to see that any reasonable solution will be irreducible. Therefore, we believe the theorem above can have many applications, we will illustrate the source location application in the following section. To take another example, in a telecommunication setting, the above theorem states that in an undirected capacitated network, any irreducible set of transmitters can transmit *concurrently* at, at least, half their maximum rates. The condition of irreducibility may be imposed by the central designer interested in the total throughput, to reduce operative costs.

Application to the above Source Location Problem. Our main corollary of Theorem 1 is the following. Call an instance of the problem ρ-regular, if all the f_i's are within a ρ-multiplicative factor of each other.

Corollary 1. *For ρ-regular instances, there is a 2ρ-approximation to the source location problem.*

Proof. The algorithm is extremely simple: starting with S, keep on deleting vertices in any order as long as the deletion doesn't decrease the total flow to t, ending with a subset $X \subseteq S$. Now, by the nature of this procedure, $f(X) = f(S)$, and also X is irreducible

as deleting any more vertices decreases the flow. We can now appeal to Theorem 1 to get that $f(S) = f(X) \geq \frac{1}{2}\sum_{i \in X} f_i \geq \frac{1}{2}|X|f_{min}$, where $f_{min} = \min_i f_i$. This gives $|X| \leq 2f(S)/f_{min}$. However, any solution X^* with $f(X^*) = f(T)$ satisfies $|X^*|f_{max} \geq \sum_{i \in X^*} f_i \geq f(S)$, implying $|X^*| \geq f(S)/f_{max} \geq |X| \cdot \frac{f_{min}}{2f_{max}}$. The proof follows from the ρ-regularity assumption.

We remark that *even with this regularity assumption.*, the directed version is inapproximable to within a factor of $o(\log n)$.[1] Furthermore, our approximation factor is optimal (assuming the unique games conjecture [10]): source location on undirected regular instances captures the *vertex cover problem* in regular graphs, which is inapproximable to within a factor of $2 - \varepsilon$, assuming the unique games conjecture [10,6].

Proof Technique. How would one prove a theorem as Theorem 1? Arguably, one needs a handle on the structure of the cuts separating a subset of terminals from a given vertex. In undirected graphs the structure of cuts has been extensively investigated. For instance, there exist cactus representations for all the minimum cuts of the graph, and the Gomory-Hu tree captures the local edge connectivities of the graph. However, we are unaware of structural results capturing cuts separating a *set* of terminals from a sink.

Another syntactic way would be to deduce inequalities involving the flow function for various subsets of the irreducible set and combine them to obtain the stated result. For instance, suppose $|X| = 3$. Then, essentially, there are a constant number of types of vertices and edges, depending on which cuts they appear in. Subsequently, one can write inequalities capturing the cut conditions, and can obtain Theorem 1 for this special case. In fact, it may be a illuminating exercise for the reader to try this out. However, for larger sets, although this process is possible, it may not be feasible to do in a 'brute force' manner.

Our approach can be thought of as a clever way of performing the above 'inequality setting'. We first show a mapping from *every* graph on to the k-dimensional hypercube, where k is the number of terminals. This mapping is 'cut-expanding': for every subset of terminals, the min-cut separating their images from the image of the sink is larger than that in the original graph. Furthermore, and this is the non-trivial part, certain min-cuts remain unchanged. This includes, in particular, the cut separating all the terminals from the sink. With this mapping, we show that the theorem need only be proved for the 'hypercube networks' that we construct. Our mapping is very similar to those used to generate what are called *mimicking networks*, and was first described by [7]. Once we go to the hypercube network, we show that the setting up of inequalities can be performed easily.

2 Proof of Theorem 1

Let G be an undirected network with an irreducible set T of terminals. We let k denote the number of terminals, that is, $|T|$. Also, for a subset $S \subseteq V(G)$, let $\delta_G(S)$ denote the capacity of the cut $(S, V(G) \setminus S)$ in G.

[1] Set cover is hard even restricted to instances with the regularity property, i.e., instances with uniform set sizes, via a simple approximation-preserving reduction from general instances.

We let Hyp_k be the graph associated with the k-dimensional Boolean hypercube $\{0,1\}^k$. The vertices of Hyp_k are the k-dimensional Boolean vectors, and there is an edge between two vectors if and only if they differ in exactly one coordinate. We let H_k denote the graph Hyp_k along with an extra vertex t^* connected to the all 1s vector, 1^k.

2.1 From Graphs to Hypercubes

We now describe a mapping $\Phi : V(G) \to V(H_k)$ along with a capacity assignment to the edges of H_k. Let $S_i \subseteq V(G)$ be the inclusion-wise minimal min-cut separating the terminal set $T_i := T \setminus i$ from t. Obviously, all $j \in T \setminus i$ lies in S_i. Note that $\delta_G(S_i) = f(T \setminus i) < f(T)$ by the irreducibility of T. Therefore, $i \notin S_i$, since otherwise the minimum cut separating T from t would be strictly less than $f(T)$, violating the max-flow-min-cut theorem. This is the place where irreducibility is crucially used.

Let S denote the inclusion-wise min-cut separating T from t. We claim that $S_i \subseteq S$ for all $i \in T$. Suppose $S_i \not\subseteq S$. The sets $S_i \cup S$ and $S_i \cap S$ respectively separate T and T_i from the sink, so $\delta_G(S_i \cup S) \geq \delta_G(S)$ and $\delta_G(S \cap S_i) > \delta_G(S_i)$. Note that the second inequality is strict, thanks to the minimality of S_i. Thus $\delta_G(S_i \cup S) + \delta_G(S_i \cap S) > \delta_G(S) + \delta_G(S_i)$, contradicting the submodularity of the cut function.

Given S and S_i's for all $i \in T$, we define the mapping Φ as follows. If $v \notin S$, then $\Phi(v) = t^*$. For $v \in S$, $\Phi(v)$ is the element of Hyp_k such that $\Phi(v)_i = 0$ if $v \in S_i$; $\Phi(v) = 1$ otherwise. Observe the following: (a) $\Phi(t) = t^*$; (b) for $i \in T$, $\Phi(i)$ is the unit vector $\mathbf{e}_i$ which has 0's in all but the ith coordinate. This follows from our previous discussion that $i \notin S_i$ but $i \in S_j$ for all $j \neq i$.

We now describe the capacities on the edges of H_k. Initially all edges have capacity 0. For each edge $(u,v) \in E(G)$ of capacity c_{uv}, we will add capacities on the edges of $E(H_k)$. If both u and v are outside S, we do nothing. If both u and v are in S, and thus $\Phi(u)$ and $\Phi(v)$ lie in $V(\mathsf{Hyp}_k)$, then we add capacity c_{uv} on all the edges of the *canonical path* between $\Phi(u)$ to $\Phi(v)$. The canonical path from x to y in Hyp_k is $x =: x_0, x_1, \ldots, x_k := y$ where x_i agrees with y on the first i coordinates, and with x in the last $(k-i)$ coordinates. Note that x_i could be the same as x_{i+1} if x and y have the same ith coordinate. If $u \in S$ and $v \notin S$, then we add a capacity c_{uv} on all edges on the canonical path from $\Phi(u)$ to 1^k, and also to the edge $(1^k, t^*)$.

To differentiate between G and H_k, given a subset X of terminals, we henceforth let $f_G(X)$ denote $f(X)$, that is, the maximum flow from X to t in G. We let $f_H(X)$ denote the maximum flow from $\Phi(X)$ to t^* in H_k with edge capacities as described above. Here, we use $\Phi(X)$ as a shorthand for $\{\Phi(x) : x \in X\}$.

Theorem 2. *Given a graph $G = (V,E)$ and an irreducible set of terminals $T \subseteq V$ of size k, the mapping $\Phi : V(G) \to V(H_k)$ as described above along with the capacity assignment on $E(H_k)$, has the following properties.*

1. *$f_G(X) \leq f_H(X)$ for all subsets $X \subseteq T$. In particular, for singletons $X = \{i\}$.*
2. *$f_G(T) = f_H(T)$.*
3. *$f_G(T') = f_H(T')$ for all subsets $T' \subseteq T$ of size $k-1$.*

Proof. 1. Consider any flow in the graph G from X to t. For any edge $(u,v) \in E(G)$ carrying positive flow, if u and v are both outside S, then $\Phi(u) = \Phi(v) = t^*$,

so we do nothing. If both are inside S, then send the same amount of flow from $\Phi(u)$ to $\Phi(v)$ along the canonical path in the hypercube. By the capacity assignment, this is a feasible flow. If exactly one of them, say u, is in S, then we use the canonical path from u to 1^k, followed by the edge $(1^k, t^*)$. This shows a feasible flow of value $f_G(X)$ from $\Phi(X)$ to t^* in H_k.

2. From part (1), it suffices to show that the capacity of the $(1^k, t^*)$ edge in H_k equals the (T, t) min-cut $\delta_G(S)$. By our construction, the $(1^k, t^*)$ gets capacity c_{uv} only for edges (u, v) with exactly one end point in S. This is precisely $\delta_G(S)$.
3. Let $T' = T \setminus i$. From part (1), it suffices to exhibit a cut in H_k separating $\Phi(T \setminus i)$ and t^* of value $f_G(T \setminus i) = \delta_G(S_i)$. We claim that the ith *dictator cut* suffices. That is, the cut separating vertices $D_i := \{x \in \mathsf{Hyp}_k \ : \ x_i = 0\}$ from the rest of the vertices in H_k. Firstly note that D_i contains $\Phi(T \setminus i)$ and t^* lies outside D_i. So this is a valid $(\Phi(T \setminus i), t^*)$ cut. Furthermore, the only edges crossing this cut belong to Hyp_k.

   Consider an edge (x, y) in Hyp_k crossing D_i with say $x_i = 0$. The capacity on this edge is contributed by edges (u, v) which have (x, y) in the canonical path from $\Phi(u)$ to $\Phi(v)$. In particular, $\Phi(u)_i = 0$ and $\Phi(v)_i = 1$; that is, $u \in S_i$ and $v \notin S_i$ and $(u, v) \in \delta_G(S_i)$. Furthermore, since this is a dictator cut, no canonical path crosses this cut more than once. In particular, the capacity of this cut is exactly the total capacity of these edges (u, v), and thus is precisely $\delta_G(S_i)$.

2.2 Bounding the Flow on the Hypercube Graph H_k

Lemma 1. $f_H(T) \geq \frac{1}{2} \sum_{i \in T} f_H(i)$.

Proof. For $1 \leq i < k$, let L_i denote the set of edges $(x, y) \in E(\mathsf{Hyp}_k)$ such that x has precisely i ones and y has $(i + 1)$ ones. Moreover, let L_k consist of the single edge $(1^k, t^*)$. We abuse notation and let L_i also denote the total capacity of the edges in L_i. Recall, $\Phi(i) = \mathbf{e}_i$. Thus the 'singleton cut' separating $\mathbf{e}_i$ from the remaining vertices is an upper bound on $f_H(i)$. Furthermore, all these singleton cuts are disjoint, and their union is $L_0 \cup L_1$. This gives

$$L_0 + L_1 \geq \sum_{i \in T} f_H(i). \tag{1}$$

Observe that for any $1 \leq i \leq k$, the edge set L_i separates t^* from $\Phi(T)$. Therefore, we get

$$L_i \geq f_H(T). \tag{2}$$

Finally, recall from the proof of (iii) of Theorem 2, that each dictator cut D_i has value $f_G(T \setminus i) < f_G(T) = f_H(T)$. Since each edge of the hypercube appears in exactly one dictator cut D_i, by adding this over all $1 \leq i \leq k$, we get

$$\sum_{0 \leq i \leq k-1} L_i \leq k \cdot f_H(T). \tag{3}$$

Using (2) for $2 \leq i \leq k - 1$, the above inequality becomes

$$L_0 + L_1 \leq 2 \cdot f_H(T). \tag{4}$$

Comparing (1) and (4) gives the lemma.

Theorem 2 and Lemma 1 imply the first part of Theorem 1.

To prove the second part, we introduce a dummy source s to G and connect it to every vertex in T with capacity of (s,i) edge being $f_i/2$. We claim that the minimum cut in this network is of value precisely $\sum_{i\in T} f_i/2$. If so, then the resulting max-flow will imply the second part of Theorem 1.

Suppose not, and let the min-cut be $(Z, V \cup s \setminus Z)$ with $s \in Z$. Let $X := T \cap Z$. Let F be the edges in $\delta(Z)$ which have endpoints in V. Let C be the total capacity of the edges in F. Since the mincut is $< \sum_{i\in T} f_i/2$, we get that $C < \sum_{i\in X} f_i/2$. However, F separates X from t, and thus the maxflow from X to t is $\leq C$. But $X \subseteq T$ is irreducible as well, and thus this violates part one of Theorem 1. To see this irreducibility of X note that if $f(X \setminus i) = f(X)$, by submodularity of f, this would imply $f(T \setminus i) = f(T)$ as well.

3 Hardness of Single Source-Sink Pair Cap-NDP

We show that the single source undirected Cap-NDP is label cover hard. The reduction is actually from the directed instances which showed label-cover hardness for directed Cap-NDP [5,3].

Consider a collection $\mathcal{G}$ of graphs obtained as follows. V consists of the following vertices. A set A of nodes partitioned into sets $A_1, \ldots, A_k$. A set B partitioned into $B_1, \ldots, B_k$. There are directed arcs of cost 0 and capacity 1 all of which are directed from some node in A_i to some node in B_j. There are nodes $a_1, \ldots, a_k$ and similarly $b_1, \ldots, b_k$. There is an arc (a_i, v) of capacity ∞ and cost C, for all $v \in A_i$. Similarly, there is an arc (v, b_j) of capacity ∞ and cost C, for all $v \in B_j$. Finally, there is an arc (s, a_i) of cost 0, capacity ∞ for all $i \in [k]$, and an arc (b_j, t) of cost 0, capacity ∞ for all $j \in [k]$. Let's call the capacity ∞ edges *big* edges. There is only one pair (s,t) with requirement R for some R (see Figure 1(a)). The reductions of [5,3] show that single source Cap-NDP is label-cover hard even on these instances.

Theorem 3 ([5,3]). *Unless* $NP \subseteq DTIME(n^{\mathrm{polylog}(n)})$, *there is no* $2^{\log^{1-\delta}(n)}$-*approximation algorithm for* Cap-NDP *for directed graphs coming from class* $\mathcal{G}$.

We now show how we obtain the hardness result for undirected graphs. If we simply make all edges undirected, the instance is not necessarily hard since the flows may travel along reverse directions. Given an undirected graph G obtained from the above instance by removing directions, we describe a simple trick that makes all capacity-1 edges (the edges between A and B) directed from left to right. This is enough for the hardness result.

Let M denote the number of capacity-1 edges. We add nodes s' and t' to V, edges $(s,s'), (t',t)$ of cost 0 and capacity $M/2$. Furthermore, we add edges (s', v) for all $v \in B$ and (t', v) for all $v \in A$. The capacity of these edges are $d(v)/2$, where $d(v)$ is the number of capacity-1 edges incident to v. The costs of all these edges are 0. Finally, we change the capacities of the capacity-1 edges to $1/2$. The demand r_{st} is set to $R + M/2$ (see Figure 1(b)).

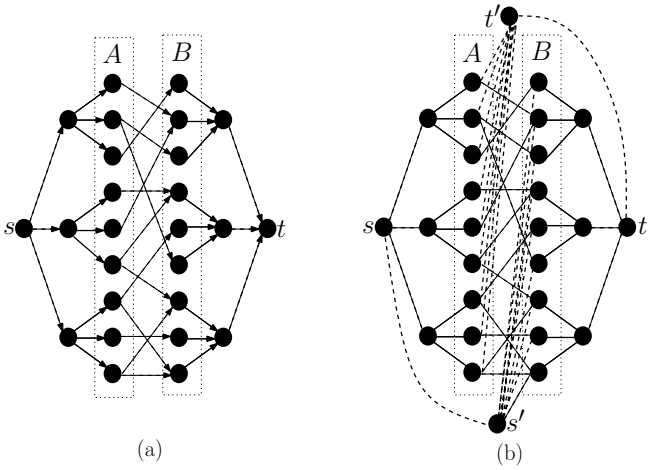


Fig. 1. The graph on the left side is the hard instance of Cap-NDP for directed graphs, and the graph on the right side is the hard instance of Cap-NDP for undirected graphs. Solid lines represent edges of cost C and dashed lines represent edges of cost 0. All edges in the left graph have capacities ∞, except for the edges from A to B, which have capacities 1. In the right graph, (s, s') and (t', t) have capacities $M/2$, where M is the number of edges between A and B. An edge between s' and $v \in B$ has capacity $d(v)/2$, where $d(v)$ is the number of edges between v and A. An edge between $v \in A$ and t' has capacity $d(v)/2$, where $d(v)$ is the number of edges between v and B. The edges between A and B have capacities $1/2$. All other edges have capacity ∞.

The intuition for the above construction is as follows. Since the capacity-$1/2$ edges (original capacity-1 edges) and the newly added edges have cost 0, we can assume they are included in the solution. With these edges, we can send $M/2$ units flow from s to t in the natural way: the flows go from s to s', then to vertices in B, then to vertices in A, to t' and finally to t. The flows use all the capacities of these edges. The remaining task is to select some other edges so that we can send R units flow in the residual graph. Notice that in the residual graph, all the capacity-$1/2$ edges are directed from left to right, with capacities 1. It is easy to see that the new added edges are useless in the residual graph. Thus, the remaining problem is equivalent to the original instance (Figure 1(a)) of Cap-NDP for directed graphs.

Now we give a more formal proof. Consider a solution to the undirected Cap-NDP. We may assume all the cost 0 edges are picked. Let F be the non-zero cost edges in the solution. Note that all of these are of the form (a_i, v) for some $v \in A_i$, or (v, b_j) for some $v \in B_j$. We abuse notation and let F also denote the corresponding arcs in the original digraph.

Claim. F, along with the 0-cost arcs, is a valid solution for the directed *Cap-NDP* instance.

Proof. Let $S \subseteq A$ be the set $\{v : (a_i, v) \in F \text{ for some } i\}$. Similarly, let $T \subseteq B$ be the set of endpoints in B neighboring to some edge in F. We claim that the edges

with one endpoint in S and the other in T, which we denote as $E(S : T)$, satisfies $|E(S : T)| \geq R$. Assuming this, we are done since the arcs are indeed directed from S to T, and since each vertex in S can receive R units of flow from s, and each vertex in T can send R units of flow to t (since these are neighboring to F), we get a feasible solution for the directed case.

Consider the following cut in the *undirected* graph with F and 0-cost edges. On the s side we have $s, \{a_1, \ldots, a_k\}, s'$ and $S \cup B \setminus T$. The t side contains the complement, that is, $t, \{b_1, \ldots, b_k\}, t'$ and $T \cup A \setminus S$. Observe there are no big edges in the cut. Big edges are either of the form (s, a_i),(t, b_j) or (a_i, v),(b_j, v). The first type are inside the s side or the t side; the second type has only F and the endpoints are made sure to be on the same side of the cut.

Therefore, the cut edges are precisely $E(S : T)$, $E(A \setminus S : B \setminus T)$ and the new edges $E(s' : T) \cup E(S : t')$. Let the capacities of these three sets be C_1 ,C_2, and C_3 respectively. Now, $C_1 = \frac{1}{2}|E(S : T)|$ is the quantity of interest, and $C_3 = \frac{1}{2}d(S) + \frac{1}{2}d(T)$, where $d(X)$ is a shorthand for $\sum_{v \in X} d(v)$. Also,

$$\begin{aligned} 2C_2 &= d(A \setminus S) - |E(A \setminus S : T)| = d(A) - d(S) - (d(T) - |E(S : T)|) \\ &= d(A) - d(S) - d(T) + |E(S : T)|. \end{aligned}$$

Thus the total capacity of this cut is $C_1 + C_2 + C_3 = \frac{1}{2}\left(d(A) + 2|E(S : T)|\right) = M/2 + |E(S : T)|$, since $d(A)$ is nothing but the number of capacity 1 edges. The capacity of the cut is $\geq M/2 + R$ since F is a feasible solution, which implies $|E(S : T)| \geq R$. Therefore, F with the 0-cost arcs form a valid solution to the directed problem as well.

The above claim, along with Theorem 3, gives the following theorem.

Theorem 4. *Unless* $NP \subseteq DTIME(n^{\mathrm{polylog}(n)})$, *there is no* $2^{\log^{1-\delta}(n)}$*-approximation for undirected, single source-sink pair* Cap-NDP.

4 Conclusion

We conclude the note with a few observations. There is a special case of Cap-NDP, which has been called the k-bipartite flow problem by [8], where given a bipartite graph with node costs and unit capacity edges, the goal is to find subsets of nodes A, B from the left and right part of minimum total cost such that the edge connectivity between A and B is at least k. In directed graphs this generalizes the densest k-subgraph problem (the version where one needs to pick the minimum number of vertices which has at least k induced edges). A similar reduction as above shows that the undirected case, and thus undirected, single source Cap-NDP, is as hard as the densest k-subgraph problem. If the goal is to just pick a min-cost subset A from one part and the set B is fixed, then a logarithmic approximation exists, and a reduction as above shows that the problem is as hard as the set cover problem.

However, one should note that the inapproximability described above only rules out unicriteria results. For instance, we haven't ruled out a solution of cost $\mathrm{polylog}(n)OPT$ which sends $\geq R/2$ flow. In fact, for the k-bipartite flow stated above, there is a solution [1] via the Räcke decomposition into trees, which obtains a solution of cost equaling OPT and sends $R/\mathrm{polylog}(n)$ flow. We think this direction may be feasible; as a

starting point we ask whether there is a $(O(\text{polylog}(n)), O(1))$-approximation for the k-bipartite flow problem.

References

1. Andreev, K., Garrod, C., Golovin, D., Maggs, B., Meyerson, A.: Simultaneous source location. Trans. on Algorithms (TALG) 6(1), 1–17 (2009) 72, 79
2. Arata, K., Iwata, S., Makino, K., Fujishige, S.: Locating sources to meet flow demands in undirected networks. J. of Algorithms 42, 54–68 (2002) 72
3. Chakrabarty, D., Chekuri, C., Khanna, S., Korula, N.: Approximability of capacitated network design. In: Günlük, O., Woeginger, G.J. (eds.) IPCO 2011. LNCS, vol. 6655, pp. 78–91. Springer, Heidelberg (2011) 72, 77
4. Dodis, Y., Khanna, S.: Design networks with bounded pairwise distance. In: ACM Symp. on Theory of Computing, STOC (1999) 72
5. Even, G., Kortsarz, G., Slany, W.: On Network Design: Fixed charge flows and the covering Steiner problem. Trans. on Algorithms (TALG) 1(1), 74–101 (2005) 72, 77
6. Feige, U.: Vertex cover is hardest to approximate on regular graphs, Tech. report, Weizmann Institute (2004) 74
7. Hagerup, T., Katajainen, J., Nishimura, N., Ragde, P.: Characterizations of multiterminal flow networks and computing flows in networks of bounded treewidth. Journal of Computer and System Sciences (JCSS) 57, 366–375 (1998) 74
8. Hajiaghayi, M.T., Khandekar, R., Kortsarz, G., Nutov, Z.: Capacitated Network Design problems: Hardness, approximation algorithms, and connections to group Steiner tree (2011), `http://arxiv.org/pdf/1108.1176.pdf` 72, 79
9. Jain, K.: A factor 2 approximation algorithm for the generalized Steiner network problem. Combinatorica 21(1), 39–60 (2001) 71, 72
10. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2-\varepsilon$. In: Proceedings of FOCS (2003) 74
11. Kortsarz, G., Nutov, Z.: A note on two source location problems. J. of Discrete Algorithms 6(3), 520–525 (2008) 72
12. Labbe, M., Peeters, D., Thisse, J.-F.: Location on networks. In: Handbook in OR and MS, vol. 8, pp. 551–624 (1995) 72
13. Sakashita, M., Makino, K., Fujishige, S.: Minimum cost source location problems with flow requirements. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, pp. 769–780. Springer, Heidelberg (2006) 72
14. Tamura, H., Sengoku, M., Shinoda, S., Abe, T.: Location problems on undirected flow networks. IEICE Trans. E(73), 1989–1993 (1990) 72
15. Tamura, H., Sengoku, M., Shinoda, S., Abe, T.: Some covering problems in location theory on flow networks. IEICE Trans. E(75), 678–683 (1992) 72
16. Wolsey, L.: An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica 2(4), 385–393 (1982) 72

Scheduling Subset Tests: One-Time, Continuous, and How They Relate

Edith Cohen[1,2], Haim Kaplan[2], and Yishay Mansour[1,3]

[1] Microsoft Research, SVC
[2] Tel Aviv University, Israel
[3] Microsoft Research, Israel
edith@cohenwang.com, {haimk,mansour}@cs.tau.ac.il

Abstract. A test scheduling instance is specified by a set of elements, a set of tests, which are subsets of elements, and numeric priorities assigned to elements. The schedule is a sequence of test invocations with the goal of covering all elements. This formulation had been used to model problems in multiple application domains from network failure detection to broadcast scheduling. The modeling considered both SUM_e and MAX_e objectives, which correspond to average or worst-case cover times over elements (weighted by priority), and both *one-time* testing, where the goal is to detect if a fault is currently present, and *continuous* testing, performed in the background in order to detect presence of failures soon after they occur. Since all variants are NP hard, the focus is on approximations.

We present combinatorial approximations algorithms for both SUM_e and MAX_e objectives on continuous and MAX_e on one-time schedules. The approximation ratios we obtain depend logarithmically on the number of elements and significantly improve over previous results. Moreover, our unified treatment of SUM_e and MAX_e objectives facilitates simultaneous approximation with respect to both.

Since one-time and continuous testing can be viable alternatives, we study their relation, which captures the overhead of continuous testing. We establish that for both SUM_e and MAX_e objectives, the ratio of the optimal one-time to continuous cover times is $O(\log n)$, where n is the number of elements. We show that this is tight as there are instances with ratio $\Omega(\log n)$. We provide evidence, however, by considering Zipf distributions, that the typical ratio is lower.

1 Introduction

An instance $(E, \mathcal{S}, \boldsymbol{p})$ of a test scheduling problem is specified by a set E of elements, a set $\mathcal{S}$ of tests, where each test is a subset of elements E, and priorities $p_e \in [0, 1]$ over elements $e \in E$. An invocation of a test $s \in \mathcal{S}$ tests all elements included in the set s and returns the faulty elements, if any exist. We seek schedules, which are sequences of tests, which cover the elements as efficiently as possible, namely, minimize the time until a fault is discovered.

We consider two objective functions: The SUM_e objectives, which minimize the prioritized sum of cover times of individual elements and MAX_e objectives, which minimize the maximum cover time of an element, weighted by priority. Operationally, we distinguish between two different modes: *one-time testing*, where the goal is to

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 81–95, 2013.

detect if an existing fault is present by initiating a sequence of tests, and *continuous testing*, performed as a background process and appropriate when failures of elements may occur any time and we would like to detect the failure soon after it occurs. For the continuous testing we consider the worst or average case over time, and either way, its cover time is always at least that of the optimal one-time testing.

This formulation naturally extends the classic set cover problem and had been used to model problems arising in different application domains. Since all variants are NP-complete, the focus is on designing approximation algorithms.

A recently studied application is silent failure detection in networks [14,12,13,6]: Elements correspond to physical or logical network elements (links, nodes, or forwarding rules in the software defined network) and tests correspond to routing paths. Invoking a test translated to sending a probe packet. Once a failure is detected, heavy-weight tools are applied to bypass or localize and correct it.

The special case of *singletons*, where each test contains a single element, was extensively studied in the context of scheduling Teletext [2], broadcast disks [1,10,3,3,3,4], and search in unstructured p2p networks [7].

One-time schedules of subset tests with respect to SUM_e objectives were studied by Feige et al [9], who gave a 4-approximation algorithm (and matching inapproximability result) (see also [5]). We recently studied stochastic and deterministic continuous schedules [6]. We gave deterministic approximation algorithms for both MAX_e and SUM_e objectives by derandomizing optimal memoryless schedules (memoryless schedules are a subclass of stochastic schedules which can be optimized by an LP or convex programs). Our work here builds on the results in [9,6] which are discussed in more detail in Section 2.

Contributions: We present novel combinatorial approximation algorithms for deterministic schedules with approximation factors $O(\log^2 n)$ for MAX_e and $O(\log n)$ for SUM_e on continuous schedules and $O(\log n)$ for MAX_e on one-time schedules (Section 3), where $n = |E|$. These ratios significantly improve over previous results [6] whose approximation ratios depend logarithmically on the number of *tests containing an element*, which can be exponential in the number of elements: Indeed, experimentally in [6], the set of tests had to be artificially restricted in order to obtain good schedules.

In some contexts, including network testing, both one-time or continuous testing are applicable, and to support informed choice, we aim to understand their relation (Section 4). Since the cover time of the optimal one-time testing is no larger than the cover time of optimal continuous testing, we are interested to study the ratio of continuous to one-time optima, which measures the overhead of continuous testing. We show that this ratio is $O(\log n)$ for both SUM_e and MAX_e objectives, and that this is tight, in the sense that some families of instances have $\Omega(\log n)$ ratios. We also give indications, by analyzing instances with priorities that are Zipf distributed, that in practice the ratio is typically lower.

In Section 5 we consider a restriction which we refer to as *choose-ℓ* scheduling, where each element must be committed to at most ℓ of the tests that include it during run-time. In Section 6 we briefly explain how to concurrently approximate both SUM_e and MAX_e objectives.

2 Preliminaries

A *testing schedule* is a sequence $\boldsymbol{\sigma}$ of tests. The sequence is infinite for continuous testing and finite for one-time testing. The *cover time* $\mathrm{T}(e, t|\boldsymbol{\sigma})$ of element e at time t by the schedule $\boldsymbol{\sigma}$ is the number of tests issued following test t until a test which includes e is invoked, that is $\mathrm{T}(e, t|\boldsymbol{\sigma}) = \min\{\Delta \geq 1 \mid e \in \sigma_{t+\Delta}\}$.

We follow notation from [6]. For a time t, $\mathsf{M}_e[t|\boldsymbol{\sigma}] = \max_e p_e \mathrm{T}(e, t|\boldsymbol{\sigma})$ is the (weighted) maximum over elements and $\mathsf{E}_e[t|\boldsymbol{\sigma}] = \sum_e p_e \mathrm{T}(e, t|\boldsymbol{\sigma})$ is the weighted sum over the elements of the cover time of e at t. The weighting, in both cases, is according to the priorities $\boldsymbol{p}$. For an element e, $\mathsf{M}_t[e|\boldsymbol{\sigma}]$ is the supremum over time t of the cover time of e at time t. We sometimes use M_t, E_e, and M_e as combinable operators which respectively take the maximum (supremum for time) or average over elements/time. When clear from context, we omit the reference to the schedule $\boldsymbol{\sigma}$ in the notation.

We study two natural objectives: *MAX*$_e$, which aims to minimize $\mathsf{M}_e\mathsf{M}_t$ the (weighted) maximum over elements, and *SUM*$_e$, which aims to minimize $\mathsf{E}_e\mathsf{M}_t$ the weighted sum over elements. For convenience, with MAX_e objectives we assume priorities are scaled so that the maximum entry is 1 and with SUM_e, they are normalized so that the sum of entries is 1. With this normalization, when $\boldsymbol{p}$ is a probability distribution over elements, SUM_e is the expected time to cover an element that is selected according to the distribution. For concreteness, we use the fault detection application for describing objectives in the sequel.

A schedule is *stochastic*, when the test σ_t invoked at each time t is determined according to a probability distribution which depends on the history $\sigma_1, \ldots, \sigma_{t-1}$. With stochastic schedules, we redefine $\mathrm{T}(e, t|\boldsymbol{\sigma})$ to be the *expected* number of steps until e is covered [6]. We append the prefix opt to an objective to denote the optimum of the objective over stochastic schedules. We append the prefix opt_D to denote the optimum of the objective over deterministic schedules.

2.1 One-Time Testing

One-time testing checks for presence of a failed element at a single specific time when we start the sequence. The schedule $\boldsymbol{\sigma}$ is finite, and we execute it until either a test detects the presence of a faulty element or to termination, when no fault is present. For element e, $\mathrm{T}(e, 0|\boldsymbol{\sigma}) = \min\{j \mid e \in \sigma_j\}$ is the cover time of e. The one-time SUM_e and MAX_e are

$$\mathrm{SUM}_e\colon \quad \mathsf{E}_e[0|\boldsymbol{\sigma}] = \sum_e p_e \mathrm{T}(e, 0|\boldsymbol{\sigma})$$

$$\mathrm{MAX}_e\colon \quad \mathsf{M}_e[0|\boldsymbol{\sigma}] = \max_e p_e \mathrm{T}(e, 0|\boldsymbol{\sigma})\ .$$

An optimal deterministic one-time schedule never performs a particular test more than once, since only the first occurrence is significant and other occurrences, if any, can only

extend the cover time of yet uncovered elements. Moreover, each test should contain at least one previously uncovered element, and therefore, an optimal schedule has length at most $n = |E|$.

A stochastic one-time schedule defines a distribution over finite deterministic sequences which is obtained by executing the stochastic schedule and stopping when all elements are covered. The maximum length of a sequence may not be bounded but the distribution is well defined for every stochastic schedule with finite expected cover times $\mathsf{E}_e[0|\boldsymbol{\sigma}]$ or $\mathsf{M}_e[0|\boldsymbol{\sigma}]$. The SUM_e cover time of the stochastic schedule is the expectation over this distribution of the SUM_e value of the deterministic sequences. Therefore, there is always a deterministic schedule with objective that is at most that expectation and opt-$\mathsf{E}_e[0] = \text{opt}_D$-$\mathsf{E}_e[0]$. Hence, there is no advantage in using a stochastic schedule when optimizing SUM_e. In contrast, it is possible for the MAX_e stochastic cover time to be lower than the deterministic optimum, that is, opt-$\mathsf{M}_e[0] \leq \text{opt}_D$-$\mathsf{M}_e[0]$. The inequality can be strict even for singleton instances: Consider a singleton instance with n equal priority elements. The optimal stochastic schedule selects a random permutation and has $\mathsf{M}_e[0] = (n+1)/2$. The optimal deterministic schedule uses a fixed permutation and has $\mathsf{M}_e[0] = n$. We discuss this further in the sequel.

Singleton Instances are fully specified by the assignment of priorities $\boldsymbol{p}$ to elements (tests). Both objectives $\mathsf{E}_e[0|\boldsymbol{\sigma}]$ and $\mathsf{M}_e[0|\boldsymbol{\sigma}]$ are minimized by the schedule $\boldsymbol{\sigma}$ that tests elements in order of decreasing priority p_i. Assuming elements are indexed by decreasing priority $p_1 \geq \cdots \geq p_n$, the optimal cover times are

$$\text{opt-}\mathsf{E}_e[0](\boldsymbol{p}) = \sum_{i=1}^{n} i p_i \tag{1}$$

$$\text{opt}_D\text{-}\mathsf{M}_e[0](\boldsymbol{p}) = \max_{i \in [n]} i p_i \ . \tag{2}$$

Subset Tests: We summarize previous results for $\mathsf{E}_e[0|\boldsymbol{\sigma}]$ and $\mathsf{M}_e[0|\boldsymbol{\sigma}]$, which establish NP hardness and approximability.

SUM_e ($\mathsf{E}_e[0|\boldsymbol{\sigma}]$): A simple greedy algorithm which sequentially selects the test that maximizes the sum of priorities of uncovered elements computes a schedule $\boldsymbol{\sigma}$ which has $\mathsf{E}_e[0|\boldsymbol{\sigma}] \leq 4\text{opt-}\mathsf{E}_e[0]$ (see, [9,5]). The problem of minimizing $\mathsf{E}_e[0|\boldsymbol{\sigma}]$ (or approximating within factor of $4 - \epsilon$ for any positive $\epsilon > 0$) is NP hard [9].

MAX_e ($\mathsf{M}_e[0|\boldsymbol{\sigma}]$): When priorities are uniform, optimizing deterministic $\mathsf{M}_e[0|\boldsymbol{\sigma}]$ is equivalent to computing a minimum set cover: The optimal $\mathsf{M}_e[0|\boldsymbol{\sigma}]$ is the size of the minimum cover. From hardness of approximation of set cover, $\mathsf{M}_e[0|\boldsymbol{\sigma}]$ is hard to approximate better than a $\ln n$ ratio [8]. When priorities are uniform, the greedy set cover algorithm guarantees a schedule with approximation ratio of $\ln n$ for $\mathsf{M}_e[0|\boldsymbol{\sigma}]$.

2.2 Continuous Testing

We first define the objectives for continuous testing. In this case the detection might start at any possible time t, and our objective considers the worse case time t.[1] Formally, for a schedule $\boldsymbol{\sigma}$ we have:

$$\mathrm{SUM}_e[\boldsymbol{\sigma}] = \mathsf{E}_e\mathsf{M}_t[\mathrm{T}(e,t|\boldsymbol{\sigma})] = \sum_e p_e \max_t \mathrm{T}(e,t|\boldsymbol{\sigma})$$
$$\mathrm{MAX}_e[\boldsymbol{\sigma}] = \mathsf{M}_e\mathsf{M}_t[\mathrm{T}(e,t|\boldsymbol{\sigma})] = \max_e \max_t p_e \mathrm{T}(e,t|\boldsymbol{\sigma}) \ .$$

With continuous testing, stochastic (expected) cover times can be lower than the corresponding optimal deterministic cover times also for both MAX_e and SUM_e [6]. Optimizing MAX_e and SUM_e over either stochastic or deterministic schedules is NP-hard.

Memoryless. A natural subclass of stochastic schedules is *memoryless schedules* [6]. Memoryless schedules are specified by a distribution $\boldsymbol{q}$ on tests, so that at each time, the invoked test is selected according to $\boldsymbol{q}$ (independently of history). For a memoryless schedule specified by $\boldsymbol{q}$ we have

$$\mathrm{SUM}_e[\boldsymbol{q}] = \sum_e \frac{p_e}{Q_e}$$
$$\mathrm{MAX}_e[\boldsymbol{q}] = \max_e \frac{p_e}{Q_e} \ ,$$

where $Q_e = \sum_{i|e\in s_i} q_i$ is the testing frequency of element e.

In [6] it was shown that the optimal memoryless cover times, with respect to either SUM_e and MAX_e, are at most twice the optimal stochastic ones. Moreover, optimal memoryless schedules can be computed efficiently, via a Linear Program (MAX_e) or a convex program (SUM_e). Optimal deterministic cover times[2] are at least the memoryless optimum, but can exceed the memoryless optimum by at most a logarithm of the number of elements and of ℓ, which we define to be the maximum number of tests which include an element. To summarize:

$$\text{opt-MAX}_e \leq \text{opt}_M\text{-MAX}_e \leq 2\text{opt-MAX}_e \tag{3}$$
$$\text{opt-SUM}_e \leq \text{opt}_M\text{-SUM}_e \leq 2\text{opt-SUM}_e \tag{4}$$
$$\text{opt}_M\text{-MAX}_e \leq \text{opt}_D\text{-MAX}_e \leq O(\log n + \log \ell)\text{opt}_M\text{-MAX}_e \tag{5}$$
$$\text{opt}_M\text{-SUM}_e \leq \text{opt}_D\text{-SUM}_e \leq O(\log \ell)\text{opt}_M\text{-SUM}_e \ . \tag{6}$$

[1] For completeness, we mention that in [6] we also considered these objectives with respect to the (limit of the) average over time t. Since the dependence on time and elements may not commute, we obtained three different objectives within each of the SUM_e ($\mathsf{M}_t\mathsf{E}_e$, $\mathsf{E}_e\mathsf{M}_t$, $\mathsf{E}_e\mathsf{E}_t$) and MAX_e ($\mathsf{M}_e\mathsf{M}_t$, $\mathsf{E}_e\mathsf{M}_t$, $\mathsf{M}_t\mathsf{E}_e$) classes. We showed that over stochastic schedules, the optimum is the same for all three variations. Over deterministic schedules, the variants that we focus on here, which take the supremum over time are the most strict and have the largest optimal cover times.

[2] For the variants we consider here, which take the supremum over time.

The optimal memoryless schedule on singleton instances has frequencies $q_e \propto p_e$ (i.e., $q_e = p_e / \sum_j p_j$) to optimize MAX_e and $q_e \propto \sqrt{p_e}$ (i.e, $q_e = \sqrt{p_e} / \sum_j \sqrt{p_j}$) to optimize SUM_e [11], where the latter is also known as Kleinrock's square-root law. The respective optima are (we use the prefix opt_M to denote the memoryless optimum):

$$\text{opt}_M\text{-MAX}_e[\boldsymbol{p}] = \max_e \frac{p_e}{q_e} = \sum_e p_e \tag{7}$$

$$\text{opt}_M\text{-SUM}_e[\boldsymbol{p}] = \sum_e \frac{p_e}{q_e} = (\sum_e \sqrt{p_e})^2 \ . \tag{8}$$

We use the following in our constructions of continuous schedules:

Lemma 1. *[3,6] For given frequencies $\boldsymbol{q}$, we can always construct a deterministic schedule so that the interval between invocations of test i is at most $2/q_i$.*

At a high level, the deterministic schedule is obtained by rounding frequencies down to integral powers of 2: $q'_i \leftarrow 2^{-\lceil \log_2 q_i \rceil}$. A set of frequencies that are integral powers of 2 that sum to at most 1 can be optimally scheduled so that each test is invoked with a period of at most $1/q'_i$ [3,6]. This is done by mapping frequencies q'_i to nodes of a binary tree so that frequency 2^{-h} is mapped to a node in depth h. The mapping is such that we prune all nodes below a mapped node. When frequencies sum to at most 1, this can always be done by mapping greedily according to decreasing h. We then construct a cyclic schedule which corresponds to traversing the leaves of the complete binary tree with depth determined by the smallest frequency. The leaves are visited by order of their reverse binary bit representation. A test is scheduled when any of the leaves under the node it is mapped to is scheduled. Lastly, we omit leaves with no mapped node above them. This can only reduce visit time.

3 Deterministic MAX_e Schedulers

We present a $O(\log n)$ approximation for one-time deterministic MAX_e scheduling and $O(\log^2 n)$ approximation for continuous deterministic MAX_e scheduling. Both algorithms use the same partition over the elements: Assuming priorities are scaled so that the largest priority is equal to 1, elements are partitioned according to the value of $\lfloor -\log_2 p_e \rfloor$, so that the set E_i for $i \geq 0$ contains all elements for which $\lfloor -\log_2 p_e \rfloor = i$. We then compute a (greedy) set cover C_i for each set E_i. Pseudo code for computing the partition and covers is in Algorithm 1. The one-time final schedule $\boldsymbol{\sigma}$ is a concatenation of the set covers C_i by increasing $i \geq 0$. (See ONETIMEMAXSCHEDULE in Algorithm 2 for pseudocode.)

Theorem 1. *Consider the one-time schedule $\boldsymbol{\sigma}$ computed by* ONETIMEMAXSCHEDULE *when the covers in* PARTITIONP2 *are computed using the greedy set cover algorithm. Then*

$$\mathsf{M}_e[0|\boldsymbol{\sigma}] \leq O(\ln n) \mathit{opt}_D\text{-}\mathsf{M}_e[0] \ .$$

Proof. We first upper bound the optimal value:

$$\text{opt-}\mathsf{M}_e[0] \leq \quad \mathsf{M}_e[0|\boldsymbol{\sigma}] \leq \max_e p_e \sum_{j \leq \lfloor -\log_2 p_e \rfloor} |C_j| \leq \quad \max_{i \geq 0} 2^{-i} \sum_{j \leq i} |C_j| \tag{9}$$

Algorithm 1. Partition elements by powers-of-2

function PARTITIONP2($\boldsymbol{p}, E$)
  $\boldsymbol{p} \leftarrow \boldsymbol{p}/\max(\boldsymbol{p})$ ▷ Scale $\boldsymbol{p}$ so that the maximum priority is 1.
  for $i \geq 0$ **do**
    $E_i \leftarrow \{e \in E_i \mid p_e \in (2^{-(i+1)}, 2^{-i}]\}$ ▷ Partition E according to priorities
  $U \leftarrow \emptyset$
  for $i \geq 0$ **do**
    $E_i \leftarrow E_i \setminus U$ ▷ remove elements already covered by higher-priority tests
    $C_i \leftarrow$SET-COVER$(E_i, \mathcal{S})$
    $U \leftarrow U \cup \{$elements covered by $C_i\}$
  return $\boldsymbol{p}, \boldsymbol{E}, \boldsymbol{C}$ ▷ $\boldsymbol{E} = \{E_i\}$ and $\boldsymbol{C} = \{C_i\}$,

We now lower bound the optimal value:

$$\begin{aligned}
\text{opt}_D\text{-}\mathsf{M}_e[0] &\geq \max_e p_e |\text{OPT-COVER}\{h \in E \mid p_h \geq p_e\}| \\
&\geq \max_{i\geq 0} 2^{-(i+1)} \max_{j\leq i} |\text{OPT-COVER}\{E_j\}| \geq \max_{i\geq 0} 2^{-(i+1)} |\text{OPT-COVER}\{E_i\}| \quad (10) \\
&\geq \max_{i\geq 0} 2^{-(i+1)} \frac{|C_i|}{\ln |E_i|} \geq \frac{1}{2\ln n} \max_{i\geq 0} 2^{-i}|C_i| \quad (11)
\end{aligned}$$

To verify (11), note that a lower bound on opt_D-$\mathsf{M}_e[0]$ is the maximum over elements e of the product of p_e by the size of the minimal set cover of all elements with priority at least p_e. For $e \in E_i$, this is lower bounded by $2^{-(i+1)}$ (the lowest possible priority of a member of E_i) times the size of the minimum set cover of E_i, which is lower bounded in turn by the size of the greedy cover $|C_i|$ divided by the worst-case approximation ratio $\ln |E_i|$.

Combining (9) and (11), to conclude the proof it suffices to establish

$$\max_{i\geq 0} 2^{-i} \sum_{j\leq i} |C_j| \leq 2 \max_{i\geq 0} 2^{-i}|C_i| \,. \quad (12)$$

Let k be the value i which maximizes $2^{-i}\sum_{j\leq i}|C_j|$. From our choice of k,

$$2^{-k} \sum_{j\leq k} |C_j| \geq 2^{-k+1} \sum_{j\leq k-1} |C_j| \,. \quad (13)$$

We expand and substitute (13) to obtain

$$\begin{aligned}
2^{-k} \sum_{j\leq k} |C_j| &= (1/2)\left(2^{-k+1} \sum_{j\leq k-1} |C_j|\right) + 2^{-k}|C_k| \\
&\leq (1/2)\left(2^{-k} \sum_{j\leq k} |C_j|\right) + 2^{-k}|C_k| \quad (14)
\end{aligned}$$

This implies that

$$2^{-k} \sum_{j\leq k} |C_j| \leq 2 \cdot 2^{-k}|C_k| \quad (15)$$

We are now ready to establish (12), using (15):

$$\max_i 2^{-i} \sum_{j \le i} |C_j| = 2^{-k} \sum_{j \le k} |C_j| \le 2 \cdot 2^{-k} |C_k| \le 2 \max_i 2^{-i} |C_i| \,. \qquad \square$$

Algorithm 2. One-Time and Continuous schedules for MAX_e

```
1: function OneTimeMaxSchedule(p, E)
2:     (p, E, C) ← PartitionP2(p, E)
3:     σ ← C_1, C_2, ...
4:     return σ
5: function ContMaxSchedule(p, E)
6:     (p, E, C) ← PartitionP2(p, E)
7:     for i ≥ 0 do
8:         for s ∈ C_i do
9:             q[s] ← 2^{-i}
10:    return ContSingletonSchedule(q / Σ_i q_[i])      ▷ Return a schedule constructed
       according to the specified frequencies as described in Lemma 1
```

To obtain a deterministic continuous-testing schedule $\boldsymbol{\sigma}$ (A Pseudocode of the construction is ContMaxSchedule in Algorithm 2), we first compute the partition and covers (PartitionP2 in Algorithm 1, using the greedy set cover algorithm). For all i, we assign frequencies 2^{-i} to the tests participating in the cover C_i and normalize so that the sum of frequencies is 1. The deterministic schedule is obtained from these frequencies by applying Lemma 1.

Theorem 2. *The schedule $\boldsymbol{\sigma}$ computed by* ContMaxSchedule *satisfies*

$$\mathit{MAX}_e[\boldsymbol{\sigma}] \le O(\ln^2 n)\mathit{opt}_D\text{-}\mathit{MAX}_e.$$

The proof of the Theorem uses the following Lemma:

Lemma 2. *The schedule $\boldsymbol{\sigma}$ computed by* ContMaxSchedule *satisfies*

$$\mathit{MAX}_e[\boldsymbol{\sigma}] \le 2 \sum_{j \ge 0} 2^{-j} |C_j| \,, \tag{16}$$

where $\boldsymbol{C}$ is the set of covers returned by PartitionP2 *(Algorithm 1).*

Proof. Consider the normalization of $\boldsymbol{q}$ in line 10 of Algorithm 2. The sum of q_i before normalization is $\sum_i 2^{-i}|C_i|$, and thus the final frequency of tests in C_i is $\frac{2^{-i}}{\sum_i 2^{-i}|C_i|}$. Consider an element $e \in E_i$. It is covered by a test s in C_j for some $j \le i$ with frequency at least $q[s] \equiv 2^{-i} / \sum_i 2^{-i}|C_i|$. The schedule $\boldsymbol{\sigma}$ invokes s at least every $2/q[s]$ steps (by Lemma 1). Therefore,

$$\mathsf{M}_t[e|\boldsymbol{\sigma}] \le 2p_e/q[s] \le 2 \cdot 2^{-i} \sum_{j \ge 0} 2^{-j}|C_j|/2^{-i} = 2 \sum_{j \ge 0} 2^{-j}|C_j| \,.$$

Since this holds for all elements, we obtain the bound for $\mathrm{MAX}_e[\boldsymbol{\sigma}] = \mathsf{M}_e\mathsf{M}_t[e|\boldsymbol{\sigma}]$.

Proof of Theorem 2. The optimum opt_D-MAX_e is lower bounded by the smallest priority in the set E_i times the size of the optimal set cover of E_i. We obtain the same lower bound we used for one-time schedules (11) in the proof of Theorem 1:

$$\begin{aligned}\text{opt}_D\text{-MAX}_e &\geq \max_{i\geq 0} 2^{-(i+1)}|\textsc{opt-cover}\{E_i\}| \\ &\geq \max_{i\geq 0} 2^{-(i+1)}|C_i|/\ln(|E_i|) \geq \frac{1}{2\ln(n)} \max_{i\geq 0} 2^{-i}|C_i| \,. \end{aligned} \tag{17}$$

By combining the upper bound in Lemma 2 and the lower bound (17), we obtain that to establish the approximation ratio of $O(\log^2 n)$, it suffices to show that for some fixed constant k,

$$\sum_{j\geq 0} 2^{-j}|C_j| \leq k\log(n) \max_{j\geq 0} 2^{-j}|C_j| \,. \tag{18}$$

To establish (18), we consider the sequence $|C_i|$, marking selected positions. We mark C_0 and then mark C_i if $|C_i| > 1.5|C_j|$, where C_j is the previously marked item. The number of marked positions is $\leq \log_{1.5} n$. This is because for all i, $|C_i| \leq |E_i| \leq n$. Consider now two consecutive marked items C_h and $C_{h'}$ where $h' > h$. We have that $|C_j| \leq 1.5|C_h|$ for every $j \in [h, h')$. Therefore, $\sum_{j=h}^{h'-1} 2^{-j}|C_j| \leq 3 \cdot 2^{-h}|C_h|$. Summing over the entire sequence we get that

$$\sum_{j\geq 0} 2^{-j}|C_j| \leq 3\log_{1.5} n \max_j 2^{-j}|C_j| \leq 6\log_2 n \max_j 2^{-j}|C_j| \,. \tag{19}$$

□

4 Relating One-Time and Continuous Testing

We study the ratio of optimal continuous to optimal one-time cover times and provide both upper and lower bounds, for both the SUM_e and MAX_e objectives. We show that the continuous deterministic optimum (and therefore also continuous stochastic and memoryless optima) is within $O(\ln m)$ of the one-time deterministic optimum, where $m \leq n$ is the number of tests in the optimal one-time schedule.

Our lower bounds use a family of instances where instance I_m has m tests and the ratio of the memoryless (continuous) optimum to the one-time deterministic optimum for instance I_m is H_m, where $H_i = \sum_{j=1}^{i} 1/j$ is the ith Harmonic number. Using the relations (3)–(6), this implies a logarithmic lower bound on the ratio also for the stochastic and deterministic continuous schedules.

4.1 Ratio for SUM_e

Theorem 3. *On any instance $I = (E, \mathcal{S}, \boldsymbol{p})$,*

$$\textit{opt}_M\textit{-SUM}_e(I) \leq \ln(m)\textit{opt-}\mathsf{E}_e[0](I) \tag{20}$$

$$\textit{opt}_D\textit{-SUM}_e(I) \leq 2\ln(m)\textit{opt-}\mathsf{E}_e[0](I) \tag{21}$$

where m is the number of tests in the optimal one-time schedule. Moreover, there is a family of instances I_i ($i \geq 1$), where instance I_i has i tests, for which

$$\frac{\textit{opt}_M\textit{-SUM}_e(I_m)}{\textit{opt-}\mathsf{E}_e[0](I_m)} = \ln(m) + O(1) \; . \tag{22}$$

Proof. Given a one-time schedule $\boldsymbol{\sigma} = s_1, \ldots, s_m$ with m tests, we construct a memoryless schedule $\boldsymbol{q}$ so that $\text{SUM}_e[\boldsymbol{q}]$ is at most $\ln m$ times $\mathsf{E}_e[0|\boldsymbol{\sigma}]$. The memoryless schedule $\boldsymbol{q}$ invokes test s_i with frequency $q_i = \frac{1}{iH_m}$. For any element e, the cover time is

$$\mathsf{M}_t[e|\boldsymbol{q}] = \frac{1}{\sum_{i|e \in s_i} q_i} \leq \frac{1}{q_{\min\{i|e \in s_i\}}} = H_m \min\{i|e \in s_i\} \; .$$

To establish (20), we can see that $\text{SUM}_e[\boldsymbol{q}]$, which must be at least $\text{opt}_M\text{-SUM}_e$, is

$$\begin{aligned} \text{SUM}_e[\boldsymbol{q}] = \mathsf{E}_e\mathsf{M}_t[\boldsymbol{q}] &= \sum_e p_e \mathsf{M}_t[e|\boldsymbol{q}] = \sum_e \frac{p_e}{\sum_{i|e \in s_i} q_i} \\ &\leq H_m \sum_e p_e \min\{i|e \in s_i\} = H_m \mathsf{E}_e[0|\boldsymbol{\sigma}] \leq \ln(m) \mathsf{E}_e[0|\boldsymbol{\sigma}] \; . \end{aligned}$$

To establish (21), we construct a deterministic continuous schedule $\boldsymbol{\sigma}'$ by applying Lemma 1 with respect to frequencies q_i for s_i. The resulting schedule invokes s_i at least once every $2/q_i \geq 2^{-\lceil \log_2(iH_m) \rceil}$ consecutive steps. We obtain that for any e, the cover time is at most

$$\mathsf{M}_t[e|\boldsymbol{\sigma}'] \leq 2/\max_{i|e \in s_i} q_i \leq 2/q_{\min\{i|e \in s_i\}} = 2H_m \min\{i|e \in s_i\} \; .$$

Therefore,

$$\begin{aligned} \text{SUM}_e[\boldsymbol{\sigma}'] = \mathsf{E}_e\mathsf{M}_t[\boldsymbol{\sigma}'] &= \sum_e p_e \mathsf{M}_t[e|\boldsymbol{\sigma}'] \leq 2 \sum_e p_e H_m \min\{i|e \in s_i\} \\ &= 2H_m \mathsf{E}_e[0|\boldsymbol{\sigma}] \; . \end{aligned}$$

We now establish the second claim. For each $m > 1$, we construct a singletons instance I_m with m tests/elements with priorities $p_i = \frac{1}{i^2}\frac{1}{S_m}$, where $S_m = \sum_{j=1}^{m} 1/i^2 \leq \pi^2/6$. The optimum $\mathsf{E}_e[0]$ for this instance is attained by invoking tests by decreasing p_i and according to (1), has:

$$\text{opt-}\mathsf{E}_e[0](I_m) = \sum_i i p_i = H_m/S_m \; . \tag{23}$$

The optimal memoryless SUM_e for I_m has square-root frequencies (8) $q_i = 1/(iH_m)$:

$$\text{opt}_M\text{-SUM}_e(\boldsymbol{p}) = (\sum_e \sqrt{p_e})^2 = H_m^2/S_m \; . \tag{24}$$

Combining (23) and (24), we get the relation $\frac{\text{opt}_M\text{-SUM}_e(I_m)}{\text{opt-}\mathsf{E}_e[0](I_m)} = H_m$. □

4.2 SUM$_e$ Continuous Schedulers

The proof of (21) in Theorem 3 includes an efficient construction of a continuous deterministic schedule from a one-time schedule. We apply this in two stages to obtain an $O(\log n)$ approximate continuous SUM$_e$ scheduler. We first apply the greedy algorithm of [9,5] to obtain a 4-approximate one-time schedule $\boldsymbol{\sigma}'$. We then apply our construction, assigning frequency $1/(iH_m)$ to the ith test of the one-time schedule $\boldsymbol{\sigma}'$, and then applying Lemma 1 to obtain a deterministic continuous schedule $\boldsymbol{\sigma}$ based on these frequencies.

The SUM$_e$ of the resulting continuous schedule is at most $O(\log n)$ times $\mathsf{E}_e[0]$ of the original one-time schedule. Now recall that opt-$\mathsf{E}_e[0] \leq$opt-SUM$_e$. We obtain the following:

Theorem 4. *The construction we outlined produces a deterministic continuous schedule $\boldsymbol{\sigma}$ so that*

$$\mathit{SUM}_e[\boldsymbol{\sigma}] \leq O(\ln n)\mathit{opt\text{-}SUM}_e.$$

4.3 Ratio for MAX$_e$

Theorem 5. *On any instance $I = (E, \mathcal{S}, \boldsymbol{p})$,*

$$\mathit{opt}_D\text{-}\mathit{MAX}_e(I) \leq O(\ln(m))\mathit{opt}_D\text{-}\mathsf{M}_e[0](I) \tag{25}$$

where m is the number of tests in the optimal one-time sequence. Moreover, there is a family of instances I_i ($i \geq 1$), where instance I_i has i tests, for which

$$\frac{\mathit{opt}_M\text{-}\mathit{MAX}_e(I_m)}{\mathit{opt}_D\text{-}\mathsf{M}_e[0](I_m)} = \ln(m) + O(1)\ . \tag{26}$$

Proof. Consider the output of PARTITIONP2 (Algorithm 1) when used with an optimal set cover subroutine. From (10), we obtain the lower bound:

$$\text{opt}_D\text{-}\mathsf{M}_e[0] \geq \max_e p_e|\text{OPT-COVER}\{h \in E \mid p_h \geq p_e\}| \geq \max_i 2^{-(i+1)}|C_i|\ . \tag{27}$$

Consider a continuous schedule $\boldsymbol{\sigma}$ computed by CONTMAXSCHEDULE (Algorithm 2) when PARTITIONP2 (Algorithm 1) is used with an optimal set cover subroutine. From Lemma 2, we have

$$\text{MAX}_e[\boldsymbol{\sigma}] \leq 2\sum_{j\geq 0} 2^{-j}|C_j|$$

Using (19) we have

$$\frac{\text{opt}_D\text{-MAX}_e}{\text{opt-}\mathsf{M}_e[0]} \leq \frac{2\sum_{j\geq 0} 2^{-j}|C_j|}{\max_i 2^{-(i+1)}|C_i|} \leq 6\log_2 n\ .$$

which establishes (25).

We construct a family of singletons instances, where instance I_n has n elements/tests and element i has priority $p_i = 1/i$. The optimal one-time schedule includes tests by decreasing priority p_i and according to (2) has opt-$\mathsf{M}_e[0](I_n) = \max_i ip_i = 1$. The optimal memoryless schedule uses $q_i \propto p_i$ and from (7) has opt$_M$-MAX$_e(I_n) = \sum_i p_i = H_n$. □

The bound on the ratio of the continuous to one-time optima also holds for stochastic schedules:

Lemma 3.

$$\textit{opt-MAX}_e = O(\log n)\textit{opt}_M\textit{-}\mathsf{M}_e[0]$$

Proof. The proof follows that of deterministic schedules (25), but instead of using integral set cover, we work with fractional covers. We define C_i to be the optimal fractional cover of elements in E_i and define $|C_i|$ to be its size (sum of fractions). We combine the upper bound opt-$\mathsf{M}_e[0] \geq \frac{1}{2}\max_i 2^{-i+1}|C_i|$ on the one-time optimum with an upper bound of $2\sum_{j\geq 0} 2^{-j}|C_j|$ on the continuous optimum, noting that Inequality (19) holds. □

4.4 Singletons with Zipf Priorities

We study the ratio for singleton instances with Zipf priorities. The instance $I_{m,\alpha}$ is specified by the number of elements/tests m and the parameter α, where the priority of element i is $p_i \propto i^{-\alpha}$. In order to understand the "typical" ratio, we examine it for general α. Recall that the ratio is $\Theta(\ln m)$ for SUM$_e$ when $\alpha = 2$ and for MAX$_e$ when $\alpha = 1$. We show that for $\alpha \neq 2$, the ratio for SUM$_e$ is constant, and similarly, for $\alpha \neq 1$, the ratio for MAX$_e$ is constant.

Ratio for SUM$_e$ when $\alpha \neq 2$. We use the expressions (1) for the one-time optimum opt-$\mathsf{E}_e[0]$ and (8) for the memoryless optimum:

$$\begin{aligned}
\frac{\text{opt}_M\text{-SUM}_e(I_{m,\alpha})}{\text{opt-}\mathsf{E}_e[0](I_{m,\alpha})} &= \frac{(\sum_{i=1}^m \sqrt{p_i})^2}{\sum_{i=1}^m ip_i} = \frac{(\sum_{i=1}^m i^{-\alpha/2})^2}{\sum_{i=1}^m i^{1-\alpha}} \\
&\leq \frac{(1+\int_1^m x^{-\alpha/2}dx)^2}{1+\int_2^m x^{1-\alpha}dx} \\
&= \frac{(1+\frac{2}{2-\alpha}(m^{1-\alpha/2}-1))^2}{1+\frac{1}{2-\alpha}(m^{2-\alpha}-2^{2-\alpha})} \\
&= \frac{4m^{2-\alpha}+\alpha^2-2\alpha m^{1-\alpha/2}}{(2-\alpha)^2\left(1-2^{2-\alpha}/(2-\alpha)+m^{2-\alpha}/(2-\alpha)\right)}.
\end{aligned}$$

We take the limit of this upper bound on the ratio as $m \to \infty$. We obtain that for $\alpha < 2$, the limit is $4/(2-\alpha)$. For $\alpha > 2$ the limit is $\alpha^2/(\alpha-2)^2$.

Ratio for MAX$_e$ for $\alpha \neq 1$. We use the expressions (2) for the one-time optimum and (7) for the memoryless optimum.

$$\begin{aligned}
\frac{\text{opt}_M\text{-MAX}_e(I_{m,\alpha})}{\text{opt-}\mathsf{M}_e[0](I_{m,\alpha})} &= \frac{\sum_{i=1}^m p_i}{\max_{i\in[m]} ip_i} = \frac{\sum_{i=1}^m i^{-\alpha}}{\max_{i\in[m]} i^{1-\alpha}} \\
&\leq \frac{1+\int_1^m x^{-\alpha}dx}{\max_{i\in[m]} i^{1-\alpha}} = \frac{m^{1-\alpha}-\alpha}{(1-\alpha)\max_{i\in[m]} i^{1-\alpha}}
\end{aligned}$$

The one-time optimum is $\max_{i\in[m]} i^{1-\alpha} = 1$ (realized for $i = 1$) when $\alpha > 1$, and is $\max_{i\in[m]} i^{1-\alpha} = m^{1-\alpha}$ (realized for $i = m$) when $\alpha < 1$. The memoryless optimum is $(m^{1-\alpha} - 1)/(1 - \alpha)$ for $\alpha < 1$ and $\approx 1/(\alpha - 1)$ for $\alpha > 1$. We obtain that for large m, the ratio is $\approx \alpha/(\alpha - 1)$ for $\alpha > 1$ and $\approx 1/(1 - \alpha)$ for $\alpha < 1$.

5 Choose-ℓ Testing

Choose-ℓ continuous testing is a natural restriction, where each element has to commit to at most ℓ of the tests which include it, so that only the selected tests may cover e at run time. Continuous scheduling as we defined it is choose-m whereas the most restricted variant is choose-1.

Theorem 6. *The choose-1 MAX$_e$ or SUM$_e$ deterministic optimum is at most $O(\ln n)$ times the unrestricted respective (MAX$_e$ or SUM$_e$) deterministic optimum. Moreover, this is tight for MAX$_e$, as there are instances with choose-1 optimum that is $\Omega(\ln n)$ times the unrestricted optimum.*

Proof. For the upper bound, we observe that the analysis for the schedule constructed in the proof of Theorem 2 for MAX$_e$ and Theorem 4 for SUM$_e$ applies also in the choose-1 testing. With Theorem 2, we can associate each element with the test containing it which participates in C_i for the smallest i. With Theorem 4, starting from a deterministic one-time schedule, we associate each element to the first test that covers it in the one-time schedule. In both cases, we obtain a choose-1 schedule which has cover time (SUM$_e$ or MAX$_e$) at most $O(\ln n)$ times the unrestricted optimum.

For the lower bound, we present a family of instances where the ratio is $\Omega(\log n)$. Our instances correspond to complete binary trees, with elements corresponding to nodes and each test to a root to leaf path. Each path is labeled by the bit string of the position of the leaf. The priority of element at level i is $\propto 2^{-i}$. The optimal choose-m schedule chooses paths in reverse bit order of the leaf labels. This schedule covers a level i node every 2^i steps and optimizes MAX$_e$. If p_r is the priority of the root, the MAX$_e$ value is $\Theta(p_r)$, where d is the depth of the tree. We show that the choose-1 optimum is logarithmically larger than the unrestricted optimum. Consider an assignment of elements to one of the paths traversing them. We now mark paths and elements top-down as follows. A node and the path traversing it are marked only if the node is not included on a marked path. We use M for the set of marked nodes. For any h, at least half the nodes on level $\leq h$ are marked. This means that the sum of priorities of marked nodes is $\sum_{e\in M} p_e \geq \Omega(d)p_r$. Let q_e be the average frequency of testing the path associated with e. The sum of q_e over marked nodes is at most 1. Since MAX$_e \geq \max_{e\in M} p_e/q_e$, we obtain that for each e, $p_e \leq q_e$MAX$_e$. Summing over $e \in M$, we obtain MAX$_e \geq \sum_{e\in M} p_e / \sum_{e\in M} q_e \geq \sum_{e\in M} p_e \geq \Omega(d)p_r$. This is factor $\Omega(d)$ larger than the unrestricted MAX$_e$. □

Closer look also shows that our bounds on the ratio between optimal continuous and one-time schedules hold for choose-1 testing, that is, the one-time deterministic optimum is at least $\Omega(1/\log n)$ the continuous choose-1 deterministic optimum (for MAX$_e$ and SUM$_e$).

Using results in [6], we obtain

Corollary 1. *(i) The optimal choose-ℓ memoryless and deterministic schedules satisfy*

$$opt_D\text{-}MAX_e \leq O(\log n + \log \ell) opt_M\text{-}MAX_e$$
$$opt_D\text{-}SUM_e \leq O(\log \ell) opt_M\text{-}SUM_e \ .$$

(ii) The optimal MAX_e and SUM_e over choose-ℓ memoryless schedules is at most twice the respective stochastic optimum.

Proof. (i) is obtained by restating the upper bounds (5) and (6). (ii) is obtained by examining the proof in [6] of the relations (3) and (4) between stochastic and memoryless schedules.

Combining these results, we obtain that the choose-1 deterministic optimal MAX_e is at most $O(\log n)$ times the stochastic choose-1 optimum. The ratio for SUM_e is $O(1)$.

5.1 Choose-1 Continuous Scheduling

When we are given an assignment $S(e)$ of elements to tests, we can obtain a deterministic schedule which is $O(1)$ optimal with respect to this assignment. We associate with test i the priority $p_i = \sum_{e|S(e)=i} p_e$. We then treat tests as singletons to obtain frequencies q_i. For SUM_e, we use $q_i \propto \sum_{e|S(e)=i} p_e$ and for MAX_e, we use $q_i \propto \sqrt{\sum_{e|S(e)=i} p_e}$. We then obtain a deterministic schedule from these frequencies using Lemma 1. Such a choose-1 assignment S is implicit in the SUM_e scheduler in Theorem 4, where elements are assigned to the first test in the one-time schedule which covers them, and in the MAX_e scheduler in Theorem2 (This implicit relation is also used and detailed in the proof of Theorem 6).

6 Combinations of Objectives

To concurrently approximate MAX_e and SUM_e objectives, we can interleave two schedules optimized for the different objectives, obtaining a schedule which has at most a factor of 2 loss in the approximation quality. In practice, we can improve on that with continuous schedules, by recalling that our continuous schedulers for MAX_e and SUM_e associate frequencies with tests and construct a schedule from these frequencies. With two objectives, we take the test-wise maximum frequency (and renormalize). This again results in losing at most a factor of two in the approximation of each objective.

Acknowledgement. This research was supported in part by the Google Inter-university center for Electronic Markets and Auctions, by a grant from the Israel Science Foundation (ISF), by a grant from United States-Israel Binational Science Foundation (BSF), and the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11).

References

1. Acharya, S., Alonso, R., Franklin, M., Zdonik, S.: Broadcast disks: data management for asymmetric communication environments. In: ACM SIGMOD (1995)
2. Ammar, M., Wong, J.: On the optimality of cyclic transmission in teletext systems. IEEE Tran. Communication 35(1), 68–73 (1987)
3. Bar-Noy, A., Bhatia, R., Naor, J., Schieber, B.: Minimizing service and operation costs of periodic scheduling. Math. Oper. Res. 27(3), 518–544 (2002)
4. Bar-Noy, A., Dreizin, V., Patt-Shamir, B.: Efficient algorithms for periodic scheduling. Computer Networks 45(2), 155–173 (2004)
5. Cohen, E., Fiat, A., Kaplan, H.: Efficient sequences of trials. In: Proc. 14th ACM-SIAM Symposium on Discrete Algorithms (2003)
6. Cohen, E., Hassidim, A., Kaplan, H., Mansour, Y., Raz, D., Tzur, Y.: Probe scheduling for efficient detection of silent failures. Technical Report cs.NI/1302.0792, arXiv (2013)
7. Cohen, E., Shenker, S.: Replication strategies in unstructured peer-to-peer networks. In: Proceedings of the ACM SIGCOMM 2002 Conference (2002)
8. Feige, U.: A threshold of $\ln n$ for approximating set cover. J. Assoc. Comput. Mach. 45, 634–652 (1998)
9. Feige, U., Lovász, L., Tetali, P.: Approximating min-sum set cover. In: Jansen, K., Leonardi, S., Vazirani, V.V. (eds.) APPROX 2002. LNCS, vol. 2462, pp. 94–107. Springer, Heidelberg (2002)
10. Hameed, S., Vaidya, N.H.: Log-time algorithms for scheduling single and multiple channel data broadcast. In: Proc. of ACM/IEEE MobiCom (1997)
11. Kleinrock, L.: Queueing Systems. Computer Applications, vol. II. Wiley-Interscience, New York (1976)
12. Nguyen, H.X., Teixeira, R., Thiran, P., Diot, C.: Minimizing probing cost for detecting interface failures: Algorithms and scalability analysis. In: INFOCOM (2009)
13. Zeng, H., Kazemian, P., Varghese, G., McKeon, N.: Automatic test packet generation. In: CONEXT (2012)
14. Zheng, Q., Cao, G.: Minimizing probing cost and achieving identifiability in probe based network link monitoring. IEEE Tran. Computers 62(3), 510–523 (2013)

On the Total Perimeter of Homothetic Convex Bodies in a Convex Container*

Adrian Dumitrescu[1] and Csaba D. Tóth[2,3]

[1] Department of Computer Science, University of Wisconsin–Milwaukee, WI, USA
dumitres@uwm.edu
[2] Department of Mathematics, California State University, Northridge, CA, USA
[3] Department of Mathematics and Statistics, University of Calgary, Canada
cdtoth@acm.org

Abstract. For two convex bodies, C and D, consider a packing S of n positive homothets of C contained in D. We estimate the total perimeter of the bodies in S, denoted $\mathrm{per}(S)$, in terms of n. When all homothets of C touch the boundary of the container D, we show that either $\mathrm{per}(S) = O(\log n)$ or $\mathrm{per}(S) = O(1)$, depending on how C and D "fit together," and these bounds are the best possible apart from the constant factors. Specifically, we establish an optimal bound $\mathrm{per}(S) = O(\log n)$ unless D is a convex polygon and every side of D is parallel to a corresponding segment on the boundary of C (for short, D is *parallel to* C). When D is parallel to C but the homothets of C may lie anywhere in D, we show that $\mathrm{per}(S) = O((1 + \mathrm{esc}(S)) \log n / \log\log n)$, where $\mathrm{esc}(S)$ denotes the total distance of the bodies in S from the boundary of D. Apart from the constant factor, this bound is also the best possible.

Keywords: Convex body, perimeter, maximum independent set, homothet, traveling salesman, approximation algorithm.

1 Introduction

A finite set $S = \{C_1, \ldots, C_n\}$ of convex bodies is a *packing* in a convex body (*container*) $D \subset \mathbb{R}^2$ if the bodies $C_1, \ldots, C_n \in S$ are contained in D and they have pairwise disjoint interiors. The term *convex body* above refers to a compact convex set with nonempty interior in $\mathbb{R}^2$. The perimeter of a convex body $C \subset \mathbb{R}^2$ is denoted $\mathrm{per}(C)$, and the total perimeter of a packing S is denoted $\mathrm{per}(S) = \sum_{i=1}^{n} \mathrm{per}(C_i)$. Our interest is estimating $\mathrm{per}(S)$ in terms of n.

We start with a few immediate observations. (1) If the convex bodies in the packing S are arbitrary, then we can assume that the packing S is in fact a tiling of the container, that is, $D = \bigcup_{i=1}^{n} C_i$. It is then easy to show that $\mathrm{per}(S) \leq \mathrm{per}(D) + 2(n-1)\,\mathrm{diam}(D)$, where $\mathrm{diam}(D)$ is the diameter of D. This bound can be achieved by subdividing D into n compact convex tiles via $n-1$ near diameter segments. (2) If all bodies in S are congruent to a convex body C, then $\mathrm{per}(S) =$

* Dumitrescu is supported in part by NSF (DMS-1001667). Tóth is supported in part by NSERC (RGPIN 35586) and NSF (CCF-0830734).

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 96–109, 2013.

$n\,\mathrm{per}(C)$, and bounding $\mathrm{per}(S)$ from above reduces to the classical problem of determining the maximum number of interior-disjoint congruent copies of C that fit in D [2].

In this paper, we consider packings S that consist of positive homothets of a convex body C. We establish an easy general bound in this case.

Proposition 1. *For every pair of convex bodies, C and D, and every packing S of n positive homothets of C in D, we have* $\mathrm{per}(S) \leq \rho(C,D)\sqrt{n}$, *where $\rho(C,D)$ depends on C and D. Apart from this multiplicative constant, this bound is the best possible.*

Motivated by applications to the traveling salesman problem with neighborhoods (TSPN), we would like to bound $\mathrm{per}(S)$ in terms of n if all homothets in S touch the boundary of D (see Fig. 1). Specifically, for a pair of convex bodies, C and D, let $f_{C,D}(n)$ denote the maximum perimeter $\mathrm{per}(S)$ of a packing of n positive homothet of C in the container D, where each element of S touches the boundary of D. We would like to estimate the growth rate of $f_{C,D}(n)$ as n goes to infinity. We prove a logarithmic upper bound $f_{C,D}(n) = O(\log n)$ for every pair of convex bodies, C and D.

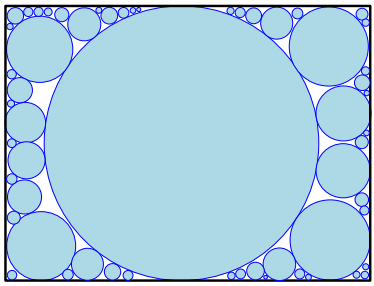


Fig. 1. A packing of disks in a rectangle container, where all disks touch the boundary of the container

Proposition 2. *For every pair of convex bodies, C and D, and every packing S of n positive homothets of C in D, where each element of S touches the boundary of D, we have* $\mathrm{per}(S) \leq \rho(C,D)\log n$, *where $\rho(C,D)$ depends on C and D.*

The upper bound $f_{C,D}(n) = O(\log n)$ is asymptotically tight for some pairs C and D, and not so tight for others. For example, it is not hard to attain an $\Omega(\log n)$ lower bound when C is an axis-aligned square, and D is a triangle (Fig. 2, left). However, $f_{C,D}(n) = \Theta(1)$ when both C and D are axis-aligned squares. We start by establishing a logarithmic lower bound in the simple setting where C is a circular disk and D is a unit square.

Theorem 1. *The total perimeter of n pairwise disjoint disks lying in the unit square $U = [0,1]^2$ and touching the boundary of U is $O(\log n)$. Apart from the constant factor, this bound is the best possible.*

We determine $f_{C,D}(n)$ up to constant factors for all pairs of convex bodies of bounded description complexity. (A planar set has *bounded description complexity* if its boundary consists of a finite number of algebraic curves of bounded degrees.) We show that either $f_{C,D} = \Theta(\log n)$ or $f_{C,D}(n) = \Theta(1)$ depending on how C and D "fit together". To distinguish these cases we need the following definitions.

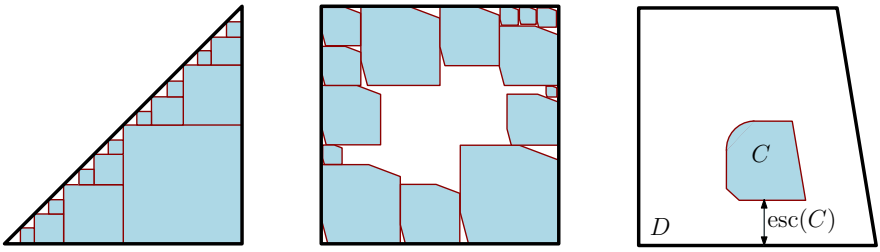


Fig. 2. Left: a square packing in a triangle where every square touches the boundary of the triangle. Middle: a packing of homothetic hexagons in a square where every hexagon touches the boundary of the square. Right: a convex body C in the interior of a trapezoid D at distance $\mathrm{esc}(C)$ from the boundary of D. The trapezoid D is *parallel to* C: every side of D is parallel and "corresponds" to a side of C.

Definitions. For a direction vector $\mathbf{d} \in \mathbb{S}^1$ and a convex body C, the *supporting line* $\ell_{\mathbf{d}}(C)$ is a directed line of direction $\mathbf{d}$ such that $\ell_{\mathbf{d}}(C)$ is tangent to C, and the closed halfplane on the left of $\ell_{\mathbf{d}}(C)$ contains C. If $\ell_{\mathbf{d}}(C) \cap C$ is a nondegenerate line segment, we refer to it as a *side* of C.

We say that a convex polygon (container) D is *parallel to* a convex body C when for every direction $\mathbf{d} \in \mathbb{S}^1$ if $\ell_{\mathbf{d}}(D) \cap D$ is a side of D, then $\ell_{\mathbf{d}}(C) \cap C$ is also a side of C. Figure 2(right) depicts a trapezoid D parallel to a convex body C. For example, every positive homothet of a convex polygon P is parallel to P; and all axis-aligned rectangles are parallel to each other.

Classification. We generalize the lower bound construction in Theorem 1 to arbitrary convex bodies, C and D, of bounded description complexity, where D is not parallel to C.

Theorem 2. *Let C and D be two convex bodies of bounded description complexity. For every packing S of n positive homothets of C in D, where each element of S touches the boundary of D, we have* $\mathrm{per}(S) \leq \rho(C, D) \log n$, *where $\rho(C, D)$ depends on C and D. Apart from the factor $\rho(C, D)$, this bound is the best possible unless D is a convex polygon parallel to C.*

If D is a convex polygon parallel to C, and every homothet of C in a packing S touches the boundary of D, then it is not difficult to see that $\mathrm{per}(S)$ is bounded.

Proposition 3. *Let C and D be convex bodies such that D is a convex polygon parallel to C. Then every packing S of n positive homothets of C in D, where each element of S touches the boundary of D, we have* $\mathrm{per}(S) \leq \rho(C, D)$, *where $\rho(C, D)$ depends on C and D.*

In the special case that D is a convex polygon parallel to C, it is also of interest to establish asymptotically tight upper bounds for $\mathrm{per}(S)$ without the assumption that the bodies in S touch the boundary of the container D. The desired dependence is in terms of n and the total distance of the bodies in S from the

boundary of D. Specifically, for two convex bodies, $C \subset D \subset \mathbb{R}^2$, let the *escape distance* $\mathrm{esc}(C)$ be the distance between C and the boundary of D (Fig. 2, right); and for a packing $S = \{C_1, \ldots, C_n\}$ in a container D, let $\mathrm{esc}(S) = \sum_{i=1}^{n} \mathrm{esc}(C_i)$. We prove the following bound for pairs of convex bodies C and D, where D is a convex polygon parallel to C.

Theorem 3. *Let C and D be two convex bodies such that D is a convex polygon parallel to C. For every packing S of n positive homothets of C in D, we have*

$$\mathrm{per}(S) \leq \rho(C, D)\,(\mathrm{per}(D) + \mathrm{esc}(S))\,\frac{\log n}{\log\log n},$$

where $\rho(C, D)$ depends on C and D. Apart from the constant factor $\rho(C, D)$, this bound is the best possible.

Motivation. In the *Euclidean Traveling Salesman Problem* (ETSP), given a set S of n points in $\mathbb{R}^d$, we wish to find a closed polygonal chain (*tour*) of minimum Euclidean length whose vertex set is S. The Euclidean TSP is known to be NP-hard, but it admits a PTAS in $\mathbb{R}^d$, where $d \in \mathbb{N}$ is constant [1]. In the *TSP with Neighborhoods* (TSPN), given a set of n sets (neighborhoods) in $\mathbb{R}^d$, we wish to find a closed polygonal chain of minimum Euclidean length that has a vertex in each neighborhood. The neighborhoods are typically simple geometric objects (of bounded description complexity) such as disks, rectangles, line segments, or lines. Since ETSP is NP-hard, TSPN is also NP-hard. TSPN admits a PTAS for certain types of neighborhoods [10], but is hard to approximate for others [4].

For n connected (possibly overlapping) neighborhoods in the plane, TSPN can be approximated with ratio $O(\log n)$ by an algorithm of Mata and Mitchell [9]. See also the survey by Bern and Eppstein [3] for a short outline of this algorithm. At its core, the $O(\log n)$-approximation relies on the following early result by Levcopoulos and Lingas [8]: every (simple) rectilinear polygon P with n vertices, r of which are reflex, can be partitioned into rectangles of total perimeter $O(\mathrm{per}(P) \log r)$ in $O(n \log n)$ time.

One approach to approximate TSPN (in particular, it achieves a constant-ratio approximation for unit disks) is the following [5,7]. Given a set S of n neighborhoods, compute a maximal subset $I \subseteq S$ of pairwise disjoint neighborhoods (i.e., a packing), compute a good tour for I, and then augment it by traversing the boundary of each set in I. Since each neighborhood in $S \setminus I$ intersects some neighborhood in I, the augmented tour visits all members of S. This approach is particularly appealing since good approximation algorithms are often available for pairwise disjoint neighborhoods [10]. The bottleneck of this approach is the length increase incurred by extending a tour of I by the total perimeter of the neighborhoods in I. An upper bound $\mathrm{per}(I) = o(\mathrm{OPT}(I) \log n)$ would immediately imply an improved $o(\log n)$-factor approximation ratio for TSPN.

Theorem 2 confirms that this approach cannot beat the $O(\log n)$ approximation ratio for most types of neighborhoods (e.g., circular disks). In the current formulation, Proposition 2 yields the upper bound $\mathrm{per}(I) = O(\log n)$ assuming a

convex container, so in order to use this bound, a tour of I needs to be augmented into a convex partition; this may increase the length by a $\Theta(\log n/\log\log n)$-factor in the worst case [6,8]. For convex polygonal neighborhoods, the bound $\mathrm{per}(I) = O(1)$ in Proposition 3 is applicable after a tour for I has been augmented into a convex partition with *parallel* edges (e.g., this is possible for axis-aligned rectangle neighborhoods, and an axis-aligned approximation of the optimal tour for I). The convex partition of a polygon with $O(1)$ distinct orientations, however, may increase the length by a $\Theta(\log n)$-factor in the worst case [8]. Overall our results confirm that we cannot beat the current $O(\log n)$ ratio for TSPN for any type of homothetic neighborhoods if we start with an arbitrary independent set I and an arbitrary near-optimal tour for I.

An improved approximation for TSPN may require additional properties of I or the initial tour for I. Alternatively, it may not be necessary to traverse the entire perimeter of all elements in I to obtain a tour for S. The escape distance $\mathrm{esc}(C)$ is a tool for measuring the necessary detour to visit a neighborhood $C \in S \setminus I$. Theorem 3 indicates that the total perimeter $\mathrm{per}(I')$ of a *second* independent set $I' \subset S \setminus I$ may be significantly larger than $\mathrm{per}(I)$.

2 Preliminaries: A Few Easy Pieces

Proof of Proposition 1. Let $\mu_i > 0$ denote the homothety factor of C_i, i.e., $C_i = \mu_i C$, for $i = 1, \ldots, n$. Since S is a packing we have $\sum_{i=1}^n \mu_i^2 \mathrm{area}(C) \le \mathrm{area}(D)$. By the Cauchy-Schwarz inequality we have $(\sum_{i=1}^n \mu_i)^2 \le n \sum_{i=1}^n \mu_i^2$. It follows that

$$\begin{aligned}\mathrm{per}(S) &= \sum_{i=1}^n \mathrm{per}(C_i) = \mathrm{per}(C) \sum_{i=1}^n \mu_i \\ &\le \mathrm{per}(C)\sqrt{n}\sqrt{\left(\sum_{i=1}^n \mu_i^2\right)} \le \mathrm{per}(C)\sqrt{\frac{\mathrm{area}(D)}{\mathrm{area}(C)}}\,\sqrt{n}.\end{aligned}$$

Set now $\rho(C, D) := \mathrm{per}(C)\sqrt{\mathrm{area}(D)/\mathrm{area}(C)}$, and the proof of the upper bound is complete.

For the lower bound, consider two convex bodies, C and D. Let U be a maximal axis-aligned square inscribed in D, and let μC be the largest positive homothet of C that fits into U. Note that $\mu = \mu(C, D)$ is a constant that depends on C and D only. Subdivide U into $\lceil\sqrt{n}\rceil^2$ congruent copies of the square $\frac{1}{\lceil\sqrt{n}\rceil}U$. Let S be the packing of n copies of $\frac{\mu}{\lceil\sqrt{n}\rceil}C$ (i.e., n translates), with at most one in each square $\frac{1}{\lceil\sqrt{n}\rceil}U$. The total perimeter of the packing is $\mathrm{per}(S) = n \cdot \frac{\mu}{\lceil\sqrt{n}\rceil}\mathrm{per}(C) = \Theta(\sqrt{n})$, as claimed. □

Proof of Proposition 2. Let $S = \{C_1, \ldots, C_n\}$ be a packing of n homothets of C in D where each element of S touches the boundary of D. Observe that $\mathrm{per}(C_i) \le \mathrm{per}(D)$ for all $i = 1, \ldots, n$. Partition the elements of S into subsets

as follows. For $k = 1, \ldots, \lceil \log_2 n \rceil$, let S_k denote the set of homothets C_i such that $\mathrm{per}(D)/2^k < \mathrm{per}(C_i) \leq \mathrm{per}(D)/2^{k-1}$; and let S_0 be the set of homothets C_i of perimeter less than $\mathrm{per}(D)/2^{\lceil \log_2 n \rceil}$. Then the sum of perimeters of the elements in S_0 is $\mathrm{per}(S_0) \leq n\,\mathrm{per}(D)/2^{\lceil \log_2 n \rceil} \leq \mathrm{per}(D)$ since $S_0 \subseteq S$ contains at most n elements altogether.

For $k = 1, \ldots, \lceil \log_2 n \rceil$, the diameter of each $C_i \in S_k$ is bounded above by

$$\mathrm{diam}(C_i) < \mathrm{per}(C_i)/2 \leq \mathrm{per}(D)/2^k. \tag{1}$$

Consequently, every point of a body $C_i \in S_k$ lies at distance at most $\mathrm{per}(D)/2^k$ from the boundary of D, denoted ∂D. Let R_k be the set of points in D at distance at most $\mathrm{per}(D)/2^k$ from ∂D. Then

$$\mathrm{area}(R_k) \leq \mathrm{per}(D)\,\frac{\mathrm{per}(D)}{2^k} = \frac{(\mathrm{per}(D))^2}{2^k}. \tag{2}$$

Since S consists of homothets, the area of any element $C_i \in S_k$ is bounded from below by

$$\mathrm{area}(C_i) = \mathrm{area}(C)\left(\frac{\mathrm{per}(C_i)}{\mathrm{per}(C)}\right)^2 \geq \mathrm{area}(C)\left(\frac{\mathrm{per}(D)}{2^k\,\mathrm{per}(C)}\right)^2. \tag{3}$$

By a volume argument, (2) and (3) yield

$$|S_k| \leq \frac{\mathrm{area}(R_k)}{\min_{C_i \in S_k} \mathrm{area}(C_i)} \leq \frac{(\mathrm{per}(D))^2/2^k}{\mathrm{area}(C)(\mathrm{per}(D))^2/(2^k\,\mathrm{per}(C))^2} = \frac{(\mathrm{per}(C))^2}{\mathrm{area}(C)} \cdot 2^k.$$

Since for $C_i \in S_k$, $k = 1, \ldots, \lceil \log_2 n \rceil$, we have $\mathrm{per}(C_i) \leq \mathrm{per}(D)/2^{k-1}$, it follows that

$$\mathrm{per}(S_k) \leq |S_k| \cdot \frac{\mathrm{per}(D)}{2^{k-1}} \leq 2\,\frac{(\mathrm{per}(C))^2}{\mathrm{area}(C)}\,\mathrm{per}(D).$$

Hence the sum of perimeters of all elements in S is bounded by

$$\mathrm{per}(S) = \sum_{k=0}^{\lceil \log_2 n \rceil} \mathrm{per}(S_k) \leq \left(1 + 2\,\frac{(\mathrm{per}(C))^2}{\mathrm{area}(C)}\lceil \log_2 n \rceil\right)\mathrm{per}(D),$$

as required. □

Proof of Proposition 3. Let $\rho'(C)$ denote the ratio between $\mathrm{per}(C)$ and the length of a shortest side of C. Recall that each $C_i \in S$ touches the boundary of polygon D. Since D is parallel to C, the side of D that supports C_i must contain a side of C_i. Let a_i denote the length of this side.

$$\mathrm{per}(S) = \sum_{i=1}^{n} \mathrm{per}(C_i) = \sum_{i=1}^{n} a_i\,\frac{\mathrm{per}(C_i)}{a_i} \leq \rho'(C) \sum_{i=1}^{n} a_i \leq \rho'(C)\,\mathrm{per}(D).$$

Set now $\rho(C, D) := \rho'(C)\,\mathrm{per}(D)$, and the proof is complete. □

3 Disks Touching the Boundary of a Square: Proof of Theorem 1

Let S be a set of n interior-disjoint disks in the unit square $U = [0,1]^2$ that touch the boundary of U. From Proposition 2 we deduce the upper bound $\mathrm{per}(S) = O(\log n)$, as required.

To prove the lower bound, it remains to construct a packing of $O(n)$ disks in the unit square $[-\frac{1}{2}, \frac{1}{2}] \times [0,1]$ such that every disk touches the x-axis, and the sum of their diameters is $\Omega(\log n)$. To each disk we associate its vertical *projection interval* (on the x-axis). The algorithm greedily chooses disks of monotonically decreasing radii such that (1) every diameter is $1/16^k$ for some $k \in \mathbb{N}$; and (2) if the projection intervals of two disks overlap, then one interval contains the other.

For $k = 0, 1, \ldots, \lfloor \log_{16} n \rfloor$, denote by S_k the set of disks of diameter $1/16^k$, constructed by our algorithm. We recursively allocate a set of intervals $X_k \subset [-\frac{1}{2}, \frac{1}{2}]$ to S_k, and then choose disks in S_k such that their projection intervals lie in X_k. Initially, $X_0 = [-\frac{1}{2}, \frac{1}{2}]$, and S_0 contains the disk of diameter 1 inscribed in $[-\frac{1}{2}, \frac{1}{2}] \times [0,1]$. The length of each maximal interval $I \subseteq X_k$ will be a multiple of $1/16^k$, so I can be covered by projection intervals of interior-disjoint disks of diameter $1/16^k$ touching the x-axis. Every interval $I \subseteq X_k$ will have the property that any disk of diameter $1/16^k$ whose projection interval is in I is disjoint from any (larger) disk in S_j, $j < k$.

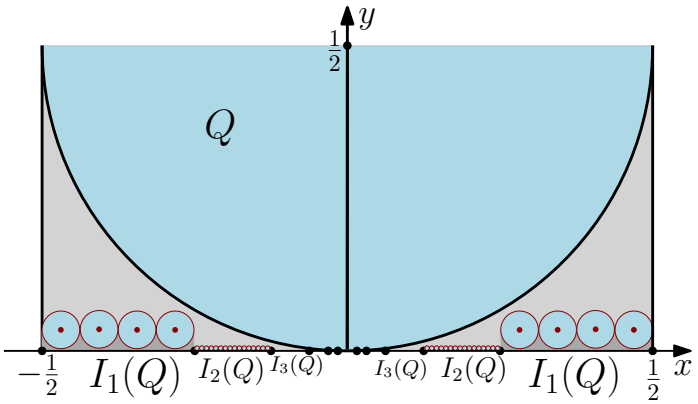


Fig. 3. Disk Q and the exponentially decreasing pairs of intervals $I_k(Q)$, $k = 1, 2, \ldots$.

Consider the disk Q of diameter 1, centered at $(0, \frac{1}{2})$, and tangent to the x-axis (see Fig. 3). It can be easily verified that:

(i) the locus of centers of disks tangent to both Q and the x-axis is the parabola $y = \frac{1}{2}x^2$; and

(ii) any disk of diameter $1/16$ and tangent to the x-axis whose projection interval is in $I_1(Q) = [-\frac{1}{2}, -\frac{1}{4}] \cup [\frac{1}{4}, \frac{1}{2}]$ is disjoint from Q.

Indeed, the center of any such disk is $(x_1, \frac{1}{16})$, for $x_1 \leq -\frac{5}{16}$ or $x_1 \geq \frac{5}{16}$, and hence lies below the parabola $y = \frac{1}{2}x^2$. Similarly, for all $k \in \mathbb{N}$, any disk of diameter $1/16^k$ and tangent to the x-axis whose projection interval is in $I_k(Q) = [-\frac{1}{2^k}, -\frac{1}{2^{k+1}}] \cup [\frac{1}{2^{k+1}}, \frac{1}{2^k}]$ is disjoint from Q. For an arbitrary disk D tangent to the x-axis, and an integer $k \geq 1$, denote by $I_k(D) \subseteq [-\frac{1}{2}, \frac{1}{2}]$ the pair of intervals corresponding to $I_k(Q)$; for $k = 0$, $I_k(D)$ consists of only one interval.

We can now recursively allocate intervals in X_k and choose disks in S_k ($k = 0, 1, \ldots, \lfloor \log_{16} n \rfloor$) as follows. Recall that $X_0 = [-\frac{1}{2}, \frac{1}{2}]$, and S_0 contains a single disk of unit diameter inscribed in the unit square $[-\frac{1}{2}, \frac{1}{2}] \times [0, 1]$. Assume that we have already defined the intervals in X_{k-1}, and selected disks in S_{k-1}. Let X_k be the union of the interval pairs $I_{k-j}(D)$ for all $D \in S_j$ and $j = 0, 1, \ldots, k-1$. Place the maximum number of disks of diameter $1/16^k$ into S_k such that their projection intervals are contained in X_k. For a disk $D \in S_j$ ($j = 0, 1, \ldots, k-1$) of diameter $1/16^j$, the two intervals in X_{k-j} each have length $\frac{1}{2} \cdot \frac{1}{2^{k-j}} \cdot \frac{1}{16^j} = \frac{8^{k-j}}{2} \cdot \frac{1}{16^k}$, so they can each accommodate the projection intervals of $\frac{8^{k-j}}{2}$ disks in S_k.

We prove by induction on k that the length of X_k is $\frac{1}{2}$, and so the sum of the diameters of the disks in S_k is $\frac{1}{2}$, $k = 1, 2, \ldots, \lfloor \log_{16} n \rfloor$. The interval $X_0 = [-\frac{1}{2}, \frac{1}{2}]$ has length 1. The pair of intervals $X_1 = [-\frac{1}{2}, -\frac{1}{4}] \cup [\frac{1}{4}, \frac{1}{2}]$ has length $\frac{1}{2}$. For $k = 2, \ldots, \lfloor \log_{16} n \rfloor$, the set X_k consists of two types of (disjoint) intervals: (a) The pair of intervals $I_1(D)$ for every $D \in S_{k-1}$ covers half of the projection interval of D. Over all $D \in S_{k-1}$, they jointly cover half the length of X_{k-1}. (b) Each pair of intervals $I_{k-j}(D)$ for $D \in S_{k-j}$, $j = 0, \ldots, k-2$, has half the length of $I_{k-j-1}(D)$. So the sum of the lengths of these intervals is half the length of X_{k-1}; although they are disjoint from X_{k-1}. Altogether, the sum of lengths of all intervals in X_k is the same as the length of X_{k-1}. By induction, the length of X_{k-1} is $\frac{1}{2}$, hence the length of X_k is also $\frac{1}{2}$, as claimed. This immediately implies that the sum of diameters of the disks in $\bigcup_{k=0}^{\lfloor \log_{16} n \rfloor} S_k$ is $1 + \frac{1}{2}\lfloor \log_{16} n \rfloor$. Finally, one can verify that the total number of disks used is $O(n)$. Write $K = \lfloor \log_{16} n \rfloor$. Indeed, $|S_0| = 1$, and $|S_k| = |X_k|/16^{-k} = 16^k/2$, for $k = 1, \ldots, K$, where $|X_k|$ denotes the total length of the intervals in X_k. Consequently, $|S_0| + \sum_{k=1}^{K} |S_k| = O(16^k) = O(n)$, as required. □

4 Homothets Touching the Boundary: Proof of Theorem 2

The upper bound $\mathrm{per}(S) = O(\log n)$ follows from Proposition 2. It remains to construct a packing S of perimeter $\mathrm{per}(S) = \Omega(\log n)$ for given C and D. Let C and D be two convex bodies with bounded description complexity. We wish to argue analogously to the case of disks in a square. Therefore, we choose an arc $\gamma \subset \partial D$ that is smooth and sufficiently "flat," but contains no side parallel to a corresponding side of C. Then we build a hierarchy of homothets of C touching the arc γ, so that the depth of the hierarchy is $O(\log n)$, and the homothety factors decrease by a constant between two consecutive levels.

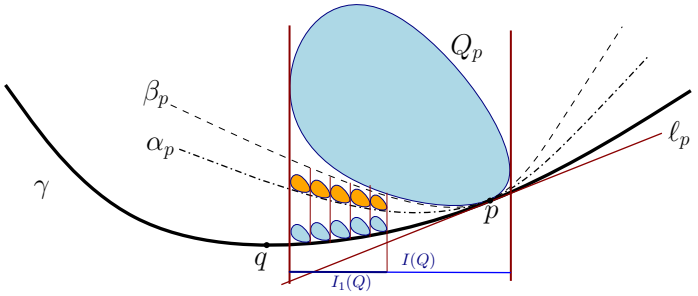


Fig. 4. If a homothet C_p is tangent to $\gamma \subset \partial D$ at point p, then there are polynomials α_p and β_p that separate γ from C_p. We can place a constant number of congruent homothets of C between α_p and β_p whose vertical projections cover $I_1(Q)$. These homothets can be translated vertically down to touch γ.

We choose an arc $\gamma \subset \partial D$ as follows. If D has a side with some direction $\mathbf{d} \in \mathbb{S}^1$ such that C has no parallel side of the same direction $\mathbf{d}$, then let γ be this side of D. Otherwise, ∂D contains an algebraic curve γ_1 of degree 2 or higher. Let $q \in \gamma_1$ be an interior point of this curve such that γ_1 is twice differentiable at q. Assume, after a rigid transformation of D if necessary, that $q = (0, 0)$ is the origin and the supporting line of D at q is the x-axis. By the inverse function theorem, there is an arc $\gamma_2 \subseteq \gamma_1$, containing q, such that γ_2 is the graph of a twice differentiable function of x. Finally, let $\gamma \subset \gamma_2$ be an arc such that the part of ∂C that has the same tangent lines as γ_2 contains no segments (sides).

For every point $p \in \gamma$, let $p = (x_p, y_p)$, and let s_p be the slope of the tangent line of D at p. Then the tangent line of D at $p \in \gamma$ is $\ell_p(x) = s_p(x - x_p)$. For any homothet Q of C, let Q_p denote a translate of Q tangent to ℓ_p at point p (Fig. 4). If both C and D have bounded description complexity, then there are constants $\rho_0 > 0$, $\kappa, \in \mathbb{N}$ and $A < B$, such that for every point $p \in \gamma$ and every homothety factor ρ, $0 < \rho < \rho_0$, the polynomials

$$\alpha_p(x) = A|x - x_p|^\kappa + s_p(x - x_p) \quad \text{and} \quad \beta_p(x) = B|x - x_p|^\kappa + s_p(x - x_p)$$

separate γ from the convex body $Q_p = (\rho C)_p$.

Similarly to the proof of Theorem 1, the construction is guided by nested *projection intervals*. Let $Q = (\rho C)_p$ be a homothet of C that lies in D and is tangent to γ at point $p \in \gamma$. Denote by $I(Q)$ the vertical projection of Q to the x-axis. For $k = 1, \ldots$, we recursively define disjoint intervals or interval pairs $I_k(Q) \subset I(Q)$ of length $|I_k(Q)| = |I(Q)|/2^k$. During the recursion, we maintain the invariant that the set $J_k(Q) = I(Q) \setminus \bigcup_{j<k} I_j(Q)$ is an interval of length $|I(Q)|/2^{k-1}$ that contains x_p. Assume that $I_1(Q), \ldots, I_{k-1}(Q)$ have been defined, and we need to choose $I_k(Q) \subset J_k(Q)$. If x_p lies in the central one quarter of $J_k(Q)$, then let $I_k(Q)$ be a pair of intervals that consists of the left and right *quarters* of $J_k(Q)$. If x_p lies to the left (right) of the central one quarter of $J_k(Q)$, then let $I_k(Q)$ be the right (left) *half* of $J_k(Q)$. It is now an easy matter to check (by induction on k) that $|x - x_p| \geq |I(Q)|/8^k$ for all $x \in I_k(Q)$.

Consequently,

$$\beta_p(x) - \alpha_p(x) \geq (B - A) \cdot \left(\frac{|I(Q)|}{8^k} \right)^{\kappa} \tag{4}$$

for all $x \in I_k(Q)$. There is a constant $\mu > 0$ such that a homothet $\mu^k Q$ with arbitrary projection interval in $I_k(Q)$ fits between the curves α_p and β_p. Refer to Fig. 4. Therefore we can populate the region between curves α_p and β_p and above $I_k(Q)$ with homothets ρQ, of homotety factors $\mu^k/2 < \rho \leq \mu^k$, such that their projection intervals are pairwise disjoint and cover $I_k(Q)$. By translating these homothets vertically until they touch γ, they remain disjoint from Q and preserve their projection intervals. We can now repeat the construction of the previous section and obtain $\lceil \log_{(2/\mu)} n \rceil$ layers of homothets touching γ, such that the total length of the projections of the homothets in each layer is $\Theta(1)$. Consequently, the total perimeter of the homothets in each layer is $\Theta(1)$, and the overall perimeter of the packing is $\Theta(\log n)$, as required. □

5 Homothets in a Parallel Container: Proof of Theorem 3

Upper bound. Let $S = \{C_1, \ldots, C_n\}$ be a packing of n homothets of a convex body C in a container D such that D is a convex polygon parallel to C. For each element $C_i \in S$, $\mathrm{esc}(C_i)$ is the distance between a side of D and a corresponding side of C_i. For each side a of D, let $S_a \subseteq S$ denote the set of $C_i \in S$ for which a is the closest side of D (ties are broken arbitrarily). Since D has finitely many sides, it is enough to show that for each side a of D, we have

$$\mathrm{per}(S_a) \leq \rho_a(C, D)\,(\mathrm{per}(D) + \mathrm{esc}(S))\ \frac{\log |S_a|}{\log \log |S_a|},$$

where $\rho_a(C, D)$ depends on a, C and D only.

Suppose that $S_a = \{C_1, \ldots, C_n\}$ is a packing of n homothets of C such that $\mathrm{esc}(C_i)$ equals the distance between C_i and side a of D. Assume for convenience that a is horizontal. Let $c \subset \partial C$ be the side of C corresponding to the side a of D. Let $\rho_1 = \mathrm{per}(C)/|c|$, and then we can write $\mathrm{per}(C) = \rho_1 |c|$. Refer to Fig. 5(left).

Denote by $b \subset c$ the line segment of length $|b| = |c|/2$ with the same midpoint as c. Since C is a convex body, the two vertical lines though the two endpoints of b intersect C in two line segments denoted h_1 and h_2, respectively. Let $\rho_2 = \min(|h_1|, |h_2|)/|b|$, and then $\min(|h_1|, |h_2|) = \rho_2 |b|$. By convexity, every vertical line that intersects segment b intersects C in a vertical segment of length at least $\rho_2 |b|$. Note that ρ_1 and ρ_2 are constants depending on C and D. For each homothet $C_i \in S_a$, let $b_i \subset \partial C_i$ be the homothetic copy of segment $b \subset \partial C$.

Put $\lambda = 2\lceil \log n / \log \log n \rceil$. Partition S_a into two subsets $S_a = S_{\mathrm{far}} \cup S_{\mathrm{close}}$ as follows. For each $C_i \in S_a$, let $C_i \in S_{\mathrm{close}}$ if $\mathrm{esc}(C_i) < \rho_2 |b_i|/\lambda$, and $C_i \in S_{\mathrm{far}}$ otherwise. For each homothet $C_i \in S_{\mathrm{close}}$, let $\mathrm{proj}_i \subseteq a$ denote the vertical projection of segment b_i onto the horizontal side a (refer to Fig. 5, right). The perimeter of each $C_i \in S_a$ is $\mathrm{per}(C_i) = \rho_1 |c_i| = 2\rho_1 |b_i| = 2\rho_1 |\mathrm{proj}_i|$. We have

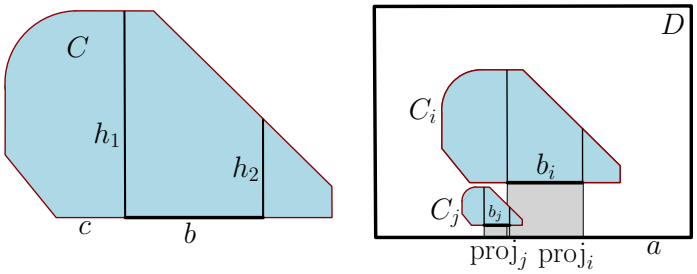


Fig. 5. Left: A convex body C with a horizontal side c. The segment $b \subset c$ has length $|b| = |c|/2$, and the vertical segments h_1 and h_2 are incident to the endpoints of b. Right: Two homothets, C_i and C_j, in a convex container D. The vertical projections of b_i and b_j onto the horizontal side a are proj_i and proj_j.

$$\mathrm{per}(S_{\mathrm{far}}) = \sum_{C_i \in S_{\mathrm{far}}} \mathrm{per}(C_i) = \sum_{C_i \in S_{\mathrm{far}}} 2\rho_1 |b_i| \leq \sum_{C_i \in S_{\mathrm{far}}} 2\rho_1 \frac{\mathrm{esc}(C_i)\,\lambda}{\rho_2}$$
$$\leq \frac{2\rho_1 \mathrm{esc}(S)}{\rho_2}\,\lambda. \tag{5}$$

It remains the estimate $\mathrm{per}(S_{\mathrm{close}})$ as an expression of λ.

$$\sum_{C_i \in S_{\mathrm{close}}} \mathrm{per}(C_i) = 2\rho_1 \sum_{C_i \in S_{\mathrm{close}}} |\mathrm{proj}_i|. \tag{6}$$

Define the *depth* function for every point of the horizontal side a by

$$d : a \to \mathbb{N}, \qquad d(x) = |\{C_i \in S_{\mathrm{close}} : x \in \mathrm{proj}_i\}|.$$

That is, $d(x)$ is the number of homothets such that the vertical projection of segment b_i contains point x. For every positive integer $k \in \mathbb{N}$, let

$$I_k = \{x \in a : d(x) \geq k\},$$

that is, I_k is the set of points of depth at least k. Since S_{close} is finite, the set $I_k \subseteq a$ is measurable. Denote by $|I_k|$ the measure (total length) of I_k. By definition, we have $|a| \geq |I_1| \geq |I_2| \geq \ldots$. A standard double counting for the integral $\int_{x \in a} d(x)\,\mathrm{d}x$ yields

$$\sum_{C_i \in S_{\mathrm{close}}} |\mathrm{proj}_i| = \sum_{k=1}^{\infty} |I_k|. \tag{7}$$

If $d(x) = k$ for some point $x \in a$, then k segments b_i, lie above x. Each $C_i \in S_{\mathrm{close}}$ is at distance $\mathrm{esc}(C_i) < \rho_2 |b_i|/\lambda$ from a. Suppose that proj_i and proj_j intersect for $C_i, C_j \in S_{\mathrm{close}}$ (Fig. 5, right). Then one of them has to be closer to a than the other: we may assume w.l.o.g. $\mathrm{esc}(C_j) < \mathrm{esc}(C_i)$. Now a vertical segment

between $b_i \subset C_i$ and $\mathrm{proj}_i \subset a$ intersects b_j. The length of this intersection segment satisfies $\rho_2|b_j| \leq \mathrm{esc}(C_i) < \rho_2|b_i|/\lambda$. Consequently, $|b_j| < |b_i|/\lambda$ (or, equivalently, $|\mathrm{proj}_j| < |\mathrm{proj}_i|/\lambda$) holds for any consecutive homothets above point $x \in a$. In particular, for the k-th smallest projection containing $x \in a$, we have $|\mathrm{proj}_k| \leq |a|/\lambda^{k-1} = |a|\lambda^{1-k}$.

We claim that

$$|I_k| \leq |a|\lambda^{\lambda-k} \qquad \text{for } k \geq \lambda + 1. \tag{8}$$

Suppose, to the contrary, that $|I_k| > |a|\lambda^{\lambda-k}$ for some $k \geq \lambda+1$. Then there are homothets $C_i \in S_{\mathrm{close}}$ of side lengths at most $|a|/\lambda^{k-1}$, that jointly project into I_k. Assuming that $|I_k| > |a|\lambda^{\lambda-k}$, it follows that the number of these homothets is at least

$$\frac{|a|\lambda^{\lambda-k}}{|a|\lambda^{1-k}} = \lambda^{\lambda-1} = \left(2\left\lceil \frac{\log n}{\log\log n}\right\rceil\right)^{2\lceil \frac{\log n}{\log\log n}\rceil - 1} > n,$$

contradicting the fact that $S_{\mathrm{close}} \subseteq S$ has at most n elements. Combining (6), (7), and (8), we conclude that

$$\begin{aligned}\mathrm{per}(S_{\mathrm{close}}) &= 2\rho_1 \sum_{k=1}^{\infty} |I_k| \leq 2\rho_1 \left(\lambda|I_1| + \sum_{k=\lambda+1}^{\infty} |I_k|\right) \leq 2\rho_1 \left(\lambda + \sum_{j=1}^{\infty} \frac{1}{\lambda^j}\right)|a| \\ &\leq 2\rho_1(\lambda+1)\,\mathrm{per}(D). \end{aligned} \tag{9}$$

Putting (5) and (9) together yields

$$\begin{aligned}\mathrm{per}(S_a) &= \mathrm{per}(S_{\mathrm{close}}) + \mathrm{per}(S_{\mathrm{far}}) \leq 2\rho_1 \left((\lambda+1)\,\mathrm{per}(D) + \frac{\mathrm{esc}(S)}{\rho_2}\,\lambda\right) \\ &\leq \rho(C,D)\,(\mathrm{per}(D) + \mathrm{esc}(S))\;\lambda = \rho(C,D)\,(\mathrm{per}(D) + \mathrm{esc}(S))\;\frac{\log n}{\log\log n},\end{aligned}$$

for a suitable $\rho(C,D)$ depending on C and D, as required; here we set $\rho(C,D) = 2\rho_1 \max(2, 1/\rho_2)$.

Lower bound for squares. We first confirm the given lower bound for squares, i.e., we construct a packing S of $O(n)$ axis-aligned squares in the unit square $U = [0,1]^2$ with total perimeter $\Omega((\mathrm{per}(U) + \mathrm{esc}(S)) \log n/\log\log n)$.

Let $n \geq 4$, and put $\lambda = \lfloor \log n/\log\log n \rfloor/2$. We arrange each square $C_i \in S$ such that $\mathrm{per}(C_i) = \lambda\,\mathrm{esc}(C_i)$. We construct S as the union of λ subsets $S = \bigcup_{j=1}^{\lambda} S_j$, where S_j is a set of congruent squares, at the same distance from the bottom side of U.

Let S_1 be a singleton set consisting of one square of side length $1/4$ (and perimeter 1) at distance $1/\lambda$ from the bottom side of U. Let S_2 be a set of 2λ squares of side length $1/(4\cdot 2\lambda)$ (and perimeter $1/(2\lambda)$), each at distance $1/(2\lambda^2)$ from the bottom side of U. Note that these squares lie strictly below the first square in S_1, since $1/(8\lambda) + 1/(2\lambda^2) < 1/\lambda$. The total length of the vertical projections of the squares in S_2 is $2\lambda \cdot 1/(8\lambda) = 1/4$.

Similarly, for $j = 3 \ldots, \lambda$, let S_j be a set of $(2\lambda)^{j-1}$ squares of side length $\frac{1}{4\cdot(2\lambda)^{j-1}}$ (and perimeter $1/(2\lambda)^{j-1}$), each at distance $1/(2^{j-1}\lambda^j)$ from the bottom side of U. These squares lie strictly below any square in S_{j-1}; and the total length of their vertical projections onto the x-axis is $(2\lambda)^{j-1} \cdot \frac{1}{4\cdot(2\lambda)^{j-1}} = 1/4$.

The number of squares in $S = \bigcup_{j=1}^{\lambda} S_j$ is

$$\sum_{j=1}^{\lambda} (2\lambda)^{j-1} = \Theta\left((2\lambda)^{\lambda}\right) = O(n).$$

The total distance from the squares to the boundary of U is

$$\mathrm{esc}(S) = \sum_{j=1}^{\lambda} (2\lambda)^{j-1} \frac{1}{2^{j-1}\lambda^j} = \lambda \frac{1}{\lambda} = 1.$$

The total perimeter of all squares in S is

$$4 \cdot \sum_{j=1}^{\lambda} \frac{1}{4} = \lambda = \Omega\left(\frac{\log n}{\log\log n}\right) = \Omega\left((\mathrm{per}(U) + \mathrm{esc}(S)) \frac{\log n}{\log\log n}\right),$$

as required.

General lower bound. We now use establish the lower bound in the general setting. Given a convex body C and a convex polygon D parallel to C, we construct a packing S of $O(n)$ positive homothets of C in D with total perimeter $\Omega((\mathrm{per}(D) + \mathrm{esc}(S)) \log n / \log\log n)$.

Let a be an arbitrary side of D. Assume w.l.o.g. that a is horizontal. Let U_C be the minimum axis-aligned square containing C. Clearly, we have $\frac{1}{2}\mathrm{per}(U_C) \leq \mathrm{per}(C) \leq \mathrm{per}(U_C)$. We first construct a packing S_U of $O(n)$ axis-aligned squares in D such that for each square $U_i \in S_U$, $\mathrm{esc}(U_i)$ equals the distance from the horizontal side a. We then obtain the packing S by inscribing a homothet C_i of C in each square $U_i \in S_U$ such that C_i touches the bottom side of U_i. Consequently, we have $\mathrm{per}(S) \geq \mathrm{per}(S_U)/2$ and $\mathrm{esc}(S) = \mathrm{esc}(S_U)$, since $\mathrm{esc}(C_i) = \mathrm{esc}(U_i)$ for each square $U_i \in S_U$.

It remains to construct the square packing S_U. Let $U(a)$ be a maximal axis-aligned square contained in D such that its bottom side is contained in a. S_U is a packing of squares in $U(a)$ that is homothetic with the packing of squares in the unit square U described previously. Put $\rho_1 = \mathrm{per}(U(a))/\mathrm{per}(U) = \mathrm{per}(U(a))/4$. We have $\mathrm{per}(S) \geq \frac{1}{4}\,\rho_1\ \Omega\left((\mathrm{per}(U) + \mathrm{esc}(S)) \frac{\log n}{\log\log n}\right)$, or

$$\mathrm{per}(S) \geq \rho(C, D) \left((\mathrm{per}(D) + \mathrm{esc}(S)) \frac{\log n}{\log\log n}\right),$$

where $\rho(C, D)$ is a factor depending on C and D, as required. □

References

1. Arora, S.: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. J. ACM 45(5), 753–782 (1998)
2. Brass, P., Moser, W.O.J., Pach, J.: Research Problems in Discrete Geometry. Springer (2005)
3. Bern, M., Eppstein, D.: Approximation algorithms for geometric problems. In: Approximation Algorithms for NP-hard Problems, pp. 296–345. PWS (1997)
4. de Berg, M., Gudmundsson, J., Katz, M.J., Levcopoulos, C., Overmars, M.H., van der Stappen, A.F.: TSP with neighborhoods of varying size. J. Algorithms 57(1), 22–36 (2005)
5. Dumitrescu, A., Mitchell, J.S.B.: Approximation algorithms for TSP with neighborhoods in the plane. J. Algorithms 48(1), 135–159 (2003)
6. Dumitrescu, A., Tóth, C.D.: Minimum weight convex Steiner partitions. Algorithmica 60(3), 627–652 (2011)
7. Dumitrescu, A., Tóth, C.D.: The traveling salesman problem for lines, balls and planes, in. In: Proc. 24th SODA, pp. 828–843. SIAM (2013)
8. Levcopoulos, C., Lingas, A.: Bounds on the length of convex partitions of polygons. In: Joseph, M., Shyamasundar, R.K. (eds.) FSTTCS 1984. LNCS, vol. 181, pp. 279–295. Springer, Heidelberg (1984)
9. Mata, C., Mitchell, J.S.B.: Approximation algorithms for geometric tour and network design problems. In: Proc. 11th SOCG, pp. 360–369. ACM (1995)
10. Mitchell, J.S.B.: A constant-factor approximation algorithm for TSP with pairwise-disjoint connected neighborhoods in the plane. In: Proc. 26th SOCG, pp. 183–191. ACM (2010)

Partial Interval Set Cover – Trade-Offs between Scalability and Optimality

Katherine Edwards[1,⋆], Simon Griffiths[2,⋆⋆], and William Sean Kennedy[3,⋆⋆⋆]

[1] Department of Computer Science
Princeton University, Princeton, NJ
ke@princeton.edu
[2] IMPA, Estrada Dona Castorina 110, Rio de Janeiro, Brasil
sgriff@impa.br
[3] Bell Labs, Murray Hill, NJ
sean.kennedy@alcatel-lucent.com

Abstract. Given an interval $I = \{1, 2, ..., n\}$ of points, a collection $\mathcal{I}$ of subintervals of I and a fraction $0 \leq r \leq 1$, we consider the following variation of partial set cover. We wish to find an optimal subset of $\mathcal{I}$ covering at least an r-fraction of I. While this problem is easily solved exactly in quadratic time using classical methods, we focus on developing scalable algorithms which return near-optimal solutions and run in near-linear time. We give a $(1+\epsilon)$-approximation algorithm running in $O(\frac{1}{\epsilon} \cdot \min\{n+|\mathcal{I}|, |\mathcal{I}| \log |\mathcal{I}|\})$ time. We also prove a tight approximation ratio of 2 for a simple greedy algorithm for this problem, improving on the bound of 9 given in [10].

1 Introduction

The explosive growth in size of modern datasets has given rise to a wealth of information; efforts to analyze such large volumes of data have presented some new computational challenges. Some datasets arising naturally from complex systems such as communication, biological, or social networks, are so big that classical algorithms offering quadratic running times can be too slow to be executed in a reasonable amount of time. It is therefore highly desirable to design algorithms which either return optimal solutions or solutions with provably small approximation ratios that scale well to large input, for example with very fast running times. In this paper we obtain a result within this paradigm. We present a near-linear time approximation algorithm for a variant of the *set cover* problem which trades off some quality of solution for greatly improved efficiency.

Given a finite universe with n elements and a collection of t subsets of these elements, the set cover problem asks for a smallest collection of subsets whose union covers the entire universe. Set cover is a classical NP-complete problem

⋆ Supported by an NSERC PGS and a Gordon Wu Fellowship.
⋆⋆ Research supported by CNPq Proc. 500016/2010-2.
⋆⋆⋆ Partially supported by NSERC PDF.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 110–125, 2013.

(see [15]) which arises frequently in optimization and analysis of data. Its diverse applications include facility location [7], machine learning [16], resource allocation [20], and data mining [9].

Johnson [14] and Lovász [19] showed that the straightforward greedy algorithm, which iteratively chooses a subset covering the maximum number of uncovered elements, returns a cover of size no more than $H_n \leq \ln n + 1$ times the size of an optimal cover.[1] Chvátal [4] strengthened this, showing that when in addition each set is given a cost and a cover of minimum cost is desired, the greedy algorithm returns a cover of total cost no more than $H_n \leq \ln n + 1$ times the cost of an optimal cover. A more precise bound was given by Slavík [22] who showed that the greedy algorithm's approximation ratio is $\ln n - \ln \ln n + \Theta(1)$. This is essentially best possible, since it is NP-hard to approximate set cover to within a multiplicative factor of $c \log n$, for some constant c (see [21], and later [1]).

In this work we investigate a natural generalization of set cover. In *partial set cover* we are given a rational number $r \in [0, 1]$ and wish to find a smallest collection of subsets covering rn elements in the universe. Since $r = 1$ is set cover, partial set cover is also NP-complete. Together with its variations, it has been well-studied in the literature (see for example [16,2,8,18,17]). The greedy approximation algorithm naturally extends to this setting and Slavík's analysis shows it has the same approximation ratio. Partial set covering has many applications, for example, data quality analysis [11,10], protein mixture identification [13], and recommendation systems [3].

In practice however, the greedy algorithm often performs very well, often choosing only a small fraction more than an optimal solution would. Gomes *et al.* [12] give experimental evidence showing the greedy algorithm on average chooses at most 7 percent more sets in the cardinality case and at most 13 percent more total cost in the weighted case than an optimal solution, both with very small variation. Unfortunately, the promise that 'usually' the algorithm returns a solution which is fixed fraction bigger than an optimal solution may be not be good enough for specific sensitive applications.

The approximation guarantee of the algorithm may not be the only issue. Algorithms with quadratic (or worse) performance guarantees can be computationally infeasible for use on extremely large datasets. Even though the greedy algorithm algorithm takes at most $O(tn)$ time, it may be useless on such large datasets. Further, such datasets may be too large to fit into memory and so the standard random access model of computation may not accurately represent the running time of the algorithm. Recently, Cormode, Karloff, and Wirth [6] observed that the greedy set cover algorithm must make many random disk accesses leading to inefficiency when the dataset is large and partially disk resident. They describe a new $(1 + c \ln n)$-approximation algorithm for set cover, for some $c > 1$ and give experimental evidence that on large datasets it greatly outperforms the standard greedy algorithm in practice.

Our focus in this article is on instances of partial set cover in which the universe is an interval (that is, the elements are ordered) and the subsets are subintervals

[1] Here H_n denotes the nth harmonic number.

of this interval. Formally, an instance of the problem consists of a positive integer n, a rational number $0 \leq r \leq 1$, an interval $[1, n] \subset \mathbb{N}$, and a set of subintervals $\mathcal{I}$, each of the form $[a, b]$ where a and b are integers between 1 and n. We refer to the integers between 1 and n as the *points*, and say that a subinterval $[a, b] \in \mathcal{I}$ *covers* the point x in $[1, n]$ if $a \leq x \leq b$. The *partial interval cover problem*, or PICP, asks to either find the minimum number of subintervals which cover at least rn of the points of $[1, n]$, or determine that no such set of subinterval exists.

Golab *et al.* [10] considered the PICP in the context of data quality assessment and knowledge discovery for ordered datasets. Examples of such natural orderings are time stamps corresponding to events in a social network, stock prices, and sequences of genome data, etc. The authors propose a method which quickly determines exceptional, such as missing, extra or out-of-order, records in an ordered dataset. The backbone of their method is an efficient method for PICP. They describe a dynamic programming approach which constructs an optimal solution in $O(n^2)$ time, where their dataset maps to the interval $[1, n]$. Furthermore, they show that the greedy algorithm can be implemented in linear time, specifically in $O(n)$ time, and give an upper bound of 9 times the size of an optimal solution for the approximation guarantee.

Our main contributions are as follows. We give a scalable algorithm for PICP which still yields a provably high quality solution. Our algorithm has an approximation ratio of $(1 + \epsilon)$ and runs in near-linear (with respect to n) time. Specifically, we prove:

Theorem 1. *There exists an algorithm which given $\epsilon > 0$, a set $\mathcal{I}$ of subintervals of $[1, n]$ and a rational number $r \in [0, 1]$ either returns $\mathcal{I}' \subseteq \mathcal{I}$ covering at least rn points of $[1, n]$ and $|\mathcal{I}'| \leq (1+\epsilon)OPT$, where OPT is the value of an optimal solution to the PICP, or determines that no such feasible solution exists, in $O(\frac{1}{\epsilon} \cdot \min\{n + |\mathcal{I}|, |\mathcal{I}| \log |\mathcal{I}|\})$ time.*

Further, improving on the result from [10] we give a tight proof that the linear time greedy algorithm in this setting has an approximation ratio of 2.

We remark that determining if an instance of PICP is feasible can be trivially done in $O(|\mathcal{I}|)$ time since we need only check if the union of all subintervals in $\mathcal{I}$ covers at least rn points. Our proof of Theorem 1 relies on two main subroutines; the first subroutine constructs a data structure which prioritizes the intervals of $\mathcal{I}$ regardless of the value of r, and the second subroutine uses the value of r to construct a near-optimal solution. The first subroutine is the source of the bottleneck in the running time of Theorem 1, whereas the second subroutine takes $O(\min\{|\mathcal{I}|, n\})$ time. Neither subroutine has a running time dependant on the value of r.

The rest of the paper is organized as follows. In Section 2, we prove a weaker version of Theorem 1; we give an $O(\min\{n+|\mathcal{I}|, |\mathcal{I}| \log |\mathcal{I}|\})$-time 2-approximation for PICP. This weakening shares many key ideas with our proof of Theorem 1, which is found in Section 3. Due to space considerations, we omit the proof of a tight approximation ratio of 2 for the greedy algorithm for PICP in this manuscript. A longer version of the paper, containing all omitted proofs is available online at `www.cs.princeton.edu/~ke/intervals.pdf`.

2 Main Idea

In this section we prove a weaker version of our main result, Theorem 1. We describe an algorithm which determines a 2-approximate solution for PICP in $O(\min\{n + |\mathcal{I}|, |\mathcal{I}| \log |\mathcal{I}|\})$ time. The ideas contained in this simpler algorithm will figure prominently in later sections and provide some intuition for our main algorithm. We begin with a preprocessing step that will allow us to make three simplifying assumptions about our input for the remainder of our discussion. For a subinterval I we denote by $left(I)$ and $right(I)$ the left and right endpoints of I, respectively.

1. The intervals in $\mathcal{I}$ are sorted in increasing order of left endpoints.
   Depending on the cardinality of $\mathcal{I}$, we use two different sorting algorithm to sort the intervals. If $|\mathcal{I}| \log |\mathcal{I}| \leq n + |\mathcal{I}|$, then we apply classical methods, for example heap sort, to sort the $|\mathcal{I}|$ intervals in $O(|\mathcal{I}| \log |\mathcal{I}|)$ time. Otherwise, we sort using a simple variant of bucket sort. We create n buckets $S_1, ..., S_n$, then for $I \in \mathcal{I}$ place interval I in bucket $S_{left(I)}$. Finally we traverse the buckets to recover the intervals in sorted order. Since there are n buckets each containing at most $|\mathcal{I}|$ intervals, this takes $O(n + |\mathcal{I}|)$ time. For further background on sorting algorithms see, for example, [5]. It follows that the input can be sorted in $O(\min\{n + |\mathcal{I}|, |\mathcal{I}| \log |\mathcal{I}|\})$ time.
2. No subinterval in $\mathcal{I}$ is contained in any other.
   Suppose there are subintervals $I \subseteq J$ in the input $\mathcal{I}$. For any feasible solution to PICP containing I, there is a corresponding feasible solution containing J of equal or lesser value so we may safely ignore I. The removal of properly contained intervals can be implemented in $O(|\mathcal{I}|)$ time on the sorted input $\mathcal{I}$. To do so, let $\mathcal{Q}$ be an initially empty queue of subintervals. Considering each subinterval $[a, b]$ in order, we first remove each subinterval $[c, d]$ on the front of $\mathcal{Q}$ for which $d < a$. Letting $[e, f]$ be the first element in $\mathcal{Q}$, if $b < f$ then we discard $[a, b]$ (since, $e < a$); otherwise, we add $[a, b]$ to Q.
3. Each point in I is covered by some subinterval in $\mathcal{I}$.
   Let P denote the set of points not covered by any interval in $\mathcal{I}$. Note that if $\frac{rn}{n-|P|} > 1$ then the instance of PICP has no feasible solution. Otherwise, we delete these $|P|$ points from $\{1, 2, ..., n\}$ and set $r \leftarrow \frac{rn}{n-|P|}$. This reduction can be implemented to run in $O(|\mathcal{I}|)$ time on input $\mathcal{I}$ satisfying Assumptions 1 and 2.

Note that after preprocessing, each point in $[1, n]$ is the left (resp., right) endpoint of at most one subinterval in $\mathcal{I}$. Thus we may unambiguously refer to the *leftmost* and *rightmost* intervals in $\mathcal{I}$.

Clearly, when $r \leq \frac{1}{n}$ solving PICP is trivial. Conversely when $\frac{n-1}{n} < r \leq 1$, the problem is simply an instance of set cover in which the sets are all intervals. For this case, we now describe a greedy algorithm which finds an optimal solution in $O(\min\{n + |\mathcal{I}|, |\mathcal{I}| \log |\mathcal{I}|\})$ time. Indeed, we can compute an optimal solution (on preprocessed input) to 1-ICP by iterating over the subintervals in order. Observe that there exists a unique subinterval $J_1 \in \mathcal{I}$ for which $left(J_1) = 1$.

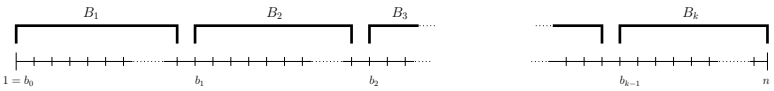


Fig. 1. Partitioning the interval into blocks

We choose it to be part of our solution and then inductively consider the problem of covering the subinterval $[right(J_1) + 1, n]$ with the subintervals whose left endpoints are greater than $right(J_1)+1$ together with the rightmost subinterval covering $right(J_1) + 1$. We leave it to the reader to verify that the solution generated is optimal, and that this algorithm can be implemented in $O(|\mathcal{I}|)$ time.

We now turn to the general case of PICP. The remainder of this section contains a proof of the following which shows how to obtain a 2-approximate solution for PICP.

Theorem 2. *There exists an algorithm which given a set $\mathcal{I}$ of subintervals of $[1, n]$ and a rational number $r \in [0, 1]$ either returns $\mathcal{I}' \subseteq \mathcal{I}$ covering at least rn points of $[1, n]$ and $|\mathcal{I}'| \leq 2OPT$, where OPT is the value of an optimal solution to the PICP, or determines that no such feasible solution exists, in $O(\frac{1}{\epsilon} \cdot \min\{n + |\mathcal{I}|, |\mathcal{I}| \log |\mathcal{I}|\})$ time.*

Our approach is the following. We partition the interval $[1, n]$ into smaller intervals (called *blocks*) $B_1, ..., B_k$ in such a way that each B_j is covered by two subintervals in $\mathcal{I}$. For each block B_j we find a subinterval of $\mathcal{I}$ which contains a maximal number of points in B_j (denoted $\mathcal{I}_j^1$) and a set of two subintervals in $\mathcal{I}$ whose union covers every point in B_j (denoted $\mathcal{I}_j^2$). Then we find a 2-approximate solution to the original instance by greedily combining the best solutions from the blocks.

2.1 Finding the Blocks

The first step of the algorithm is to find a partition of I into blocks. We choose *breakpoints* $1 = b_0, b_1, ..., b_{k-1}$ which specify blocks $B_j = [b_{j-1}, b_j - 1]$ for $1 \leq j \leq k-1$ and $B_k = [b_{k-1}, n]$ as in Figure 1. The key is to choose the breakpoints so that each block admits a bounded size solution, as guaranteed by the next lemma.

Lemma 1. *There exists a choice of breakpoints $b_0, \ldots, b_{k-1}$ such that*

1. *every subinterval of $\mathcal{I}$ contains at most one breakpoint, and*
2. *for every $1 \leq j \leq k$, there exist two subintervals in$\mathcal{I}$ whose union contains the block B_j.*

Further, such breakpoints can be computed in $O(|\mathcal{I}|)$ time.

Before computing the breakpoints, we introduce some notation.

Definition 3. For any interval $S \subseteq [1, n]$, the *restriction* of $\mathcal{I}$ to S is $\mathcal{I}|_S = \{I \cap S \neq \emptyset; I \in \mathcal{I}\}$. The *proper restriction* of $\mathcal{I}$ to S is $\mathcal{I}|_S^P = \{I \in \mathcal{I}|_S : \forall J \in \mathcal{I}|_S, I \not\subset J\}$. For each subinterval $J \in \mathcal{I}|_S^P$, a *preimage* of J is any interval $J' \in \mathcal{I}$ such that $J = J' \cap S$.

Proof (Proof of Lemma 1).

Set $b_0 = 1$. We pick $b_1, ..., b_{k-1}$ iteratively. Given $b_0, ..., b_{j-1}$ for $j \geq 1$, pick b_j as follows. First, let I_j^1 be the unique leftmost subinterval of $\mathcal{I}|_{[b_{j-1},n]}^P$. Let $i = right(I_j^1)+1$. Similarly, let I_j^2 be the leftmost subinterval of $\mathcal{I}|_{[i,n]}^P$. If $right(I_j^2)+1 < n$, then set $b_j = right(I_j^2) + 1$; otherwise, we are done.

We immediately see that each block is covered by two subintervals, namely I_j^1 and I_j^2. Moreover, the choice of the breakpoints ensures that no subinterval in $\mathcal{I}$ contains more than one breakpoint. Hence, our chosen intervals satisfy Properties 1 and 2. This process considers each subinterval at most once and therefore can be implemented in $O(|\mathcal{I}|)$ time.

Now, for each block B_j it is easy to compute $\mathcal{I}_j^1$ and $\mathcal{I}_j^2$. Indeed, we set $\mathcal{I}_j^2 = \{I_j^1, I_j^2\}$, as in the proof of Lemma 1 and to find $\mathcal{I}_j^1$, a collection of exactly one subinterval of $\mathcal{I}$ which covers a maximal number of points in B_j, we simply scan through the intervals of $\mathcal{I}|_{B_j}^P$. For convenience, we define $\mathcal{I}_j^0 = \emptyset$ for each j between 1 and k.

2.2 Greedily Combining the Partial Solutions

It remains to find a 2-approximate solution $\mathcal{S}$ by combining the partial solutions. For each block B_j we restrict ourselves to choosing either the subintervals in$\mathcal{I}_j^0$, $\mathcal{I}_j^1$ or $\mathcal{I}_j^2$. Therefore a solution can be represented as a vector $x \in \{0, 1, 2\}^k$, where for each $j = 1, ..., k$, $\mathcal{I}_j^{x_j}$ is the set of subintervals chosen from the block B_j. The *cost* of a solution x is $cost(x) = \sum_{j=1}^k x_j$. We find a vector $x^* \in \{0, 1, 2\}^k$ minimizing $cost(x)$ and subject to $\left|\bigcup_{j=1}^k \mathcal{I}_j^{x_j^*}\right| \geq rn$.

Starting with $x^0 = (0, 0, ...0)$, we use a *greedy picking* technique: for each iteration $\ell = 0, 1, ...$, find $x^{\ell+1}$ by increasing by 1 an element of x^ℓ which yields a largest possible increase in the value of $\left|\bigcup_{j=1}^k \mathcal{I}_j^{x_j^{\ell+1}}\right|$. We claim that for each iteration ℓ, x^ℓ covers as many points as any other solution whose cost is ℓ. Hence, by stopping at iteration ℓ' when $\left|\bigcup_{j=1}^k \mathcal{I}_j^{x_j^{\ell'}}\right| \geq rn$, it follows that our final solution $x^* = x^{\ell'}$ is the desired vector. This claim is a special case of Lemma 4 proven below, and so here, we only sketch the key ideas.

For any $j = 1, 2, ..., k$ and $i = 0, 1, 2$, the intervals of $\mathcal{I}_j^i$ are completed contained in B_j. Hence, we can quantify the increase of $\left|\bigcup_{j=1}^k \mathcal{I}_j^{x_j^\ell}\right|$ given by increasing the value of the j-th element in x^ℓ as follows.

Definition 4. For each $j \in 1, ..., k$ and $r < 2$, let the *marginal of* $\mathcal{I}_j^{r+1}$, denoted $g(\mathcal{I}_j^{r+1})$, be the number of points covered in B_j by the intervals of $\mathcal{I}_j^{r+1}$ minus the number of points covered in B_j by the intervals of $\mathcal{I}_j^r$, that is, $g(\mathcal{I}_j^{r+1}) = \left|\bigcup_{I \in \mathcal{I}_j^{r+1}} I\right| - \left|\bigcup_{I \in \mathcal{I}_j^r} I\right|$.

Observation 5. *Given* $x^\ell = (x_1^\ell, ..., x_t^\ell, ..., x_k^\ell)$ *where* $x_t^\ell < 2$ *and letting* $x^{\ell+1} = (x_1^\ell, ..., x_t^\ell + 1, ..., x_k^\ell)$ *we have*

$$\left|\bigcup_{j=1}^{k} \mathcal{I}_j^{x_j^{\ell+1}}\right| = \left|\bigcup_{j=1}^{k} \mathcal{I}_j^{x_j^{\ell}}\right| + g\left(\mathcal{I}_t^{x_t^{\ell}+1}\right).$$

So, the largest increase in $\left|\bigcup_{j=1}^{k} \mathcal{I}_j^{x_j^{\ell+1}}\right|$ is given by increasing some $x_j^\ell < 2$ which has maximum marginal. To prove that $x^{\ell+1}$ covers as many points as any other solution whose cost is $\ell + 1$ we show that for each block the marginals are decreasing.

Fact 6. *For each* j, $g(\mathcal{I}_j^1) \geq g(\mathcal{I}_j^2)$.

Proof. Since $\left|\bigcup_{I \in \mathcal{I}_j^0} I\right| = 0$, we have $g(\mathcal{I}_j^1) = \left|\bigcup_{I \in \mathcal{I}_j^1} I\right|$, and so, $g(\mathcal{I}_j^1) + g(\mathcal{I}_j^2) = \left|\bigcup_{I \in \mathcal{I}_j^2} I\right|$. If $g(\mathcal{I}_j^1) < g(\mathcal{I}_j^2)$ then $2\left|\bigcup_{I \in \mathcal{I}_j^1} I\right| = 2g(\mathcal{I}_j^1) < \left|\bigcup_{I \in \mathcal{I}_j^2} I\right|$. So, the longest subinterval in $\mathcal{I}_j^2$ is longer than the subinterval of $\mathcal{I}_j^1$, contradicting our choice of $\mathcal{I}_j^1$.

This suggests intuitively that it is always a disadvantage to increase an element whose corresponding marginal is not maximum and that there is no disadvantage to increase any element whose corresponding marginal is maximum. We omit further details.

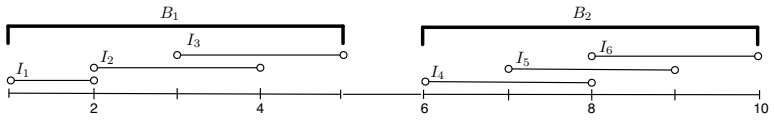


Fig. 2. Blocks B_1, B_2 on points $[1, 10]$ and containing six subintervals $I_1 = [1, 2], I_2 = [2, 4], I_3 = [3, 5], I_4 = [6, 8], I_5 = [7, 9], I_6 = [8, 10]$. A possible partial solution for B_1 is $\mathcal{I}_1^0 = \emptyset$, $\mathcal{I}_1^1 = \{I_2\}$ and $\mathcal{I}_1^2 = \{I_1, I_3\}$ and for B_2 is $\mathcal{I}_2^0 = \emptyset$, $\mathcal{I}_2^1 = \{I_4\}$ and $\mathcal{I}_2^2 = \{I_4, I_6\}$. Since $\mathcal{I}_1^0$ covers 0 points, $\mathcal{I}_1^1$ covers 3 points and $\mathcal{I}_1^2$ covers 5 points, the marginals are $g(\mathcal{I}_1^1) = 3$ and $g(\mathcal{I}_1^2) = 2$. Similarly, $g(\mathcal{I}_2^1) = 3$ and $g(\mathcal{I}_1^2) = 2$. Now, if $r = .7$ then a possible set of iterations for the greedy picking algorithm are $x^0 = (0, 0), x^1 = (1, 0), x^2 = (1, 1), x^3 = (2, 1)$. x^3 yields the final solution $\{I_1, I_3, I_5\}$, which covers 8 points.

2.3 Performance Analysis

We prove in this section that there exists a solution to PICP containing at most $2OPT$ subintervals. To do so, we show that there exists a vector $o \in \{0,1,2\}^k$ such that $cost(o) \leq 2OPT$ and $\left|\bigcup_{j=1}^{k} \mathcal{I}_j^{o_j}\right| \geq rn$. It follows that the vector $x^* \in \{0,1,2\}^k$ as found in Section 2.2 also satisfies $cost(x^*) \leq 2OPT$ and $\left|\bigcup_{j=1}^{k} \mathcal{I}_j^{x_j^*}\right| \geq rn$. We recall that the subintervals in each $\mathcal{I}_j^i$ are restrictions of subintervals to blocks. So, we can determine a solution of PICP corresponding to x^* by determining a collection $\mathcal{S}$ of preimages of the subintervals in $\bigcup_{j=1}^{k} \mathcal{I}_j^{x_j^*}$. It follows immediately that $\mathcal{S}$ is the desired solution.

To find such a vector o, we consider some optimal solution $\mathcal{I}^*$ to PICP. For $1 \leq j \leq k$, let o_j^* be the number of subintervals in $\mathcal{I}^*$ which cover only points in block B_j, and for $1 \leq j \leq k-1$, define g_j^* to be the number of subintervals in $\mathcal{I}^*$ which cover points in both B_j and B_{j+1}. Since no subinterval contains two breakpoints, we have $OPT = \sum_{j=1}^{k} o_j^* + \sum_{j=1}^{k-1} g_j^*$.

Now, let $o_1 = \min\{o_1^* + g_1^*, 2\}$, for each $2 \leq j \leq k-1$, $o_j = \min\{o_j^* + g_{j-1}^* + g_j^*, 2\}$, and $o_k = \min\{o_k^* + g_{k-1}^*, 2\}$. We have

$$\sum_{j=1}^{k} o_j \leq \sum_{j=1}^{k} o_j^* + 2\sum_{j=1}^{k-1} g_j^* \leq 2\left(\sum_{j=1}^{k} o_j^* + \sum_{j=1}^{k-1} g_j^*\right) = 2OPT.$$

To complete the proof it is enough to show that $\bigcup_{j=1}^{k} \mathcal{I}_j^{o_j}$ covers at least rn points. To do so, we show that $\bigcup_{j=1}^{k} \mathcal{I}_j^{o_j}$ covers as least as many points as the optimal solution $\mathcal{I}^*$, which covers at least rn points. Notice that

- if $o_j = 0$ then $\mathcal{I}^*$ covers no points in B_j,
- if $o_j = 1$ then exactly one subinterval in $\mathcal{I}^*$ covers any points in B_j and $\mathcal{I}_j^1$ covers as many points of B_j as any other subinterval in $\mathcal{I}$, and
- if $o_j = 2$ then $\mathcal{I}_j^2$ covers every point in B_j.

It follows that the subintervals in $\bigcup_{j=1}^{k} \mathcal{I}_j^{o_j}$ cover at least as many points as the subintervals in $\mathcal{I}^*$.

2.4 Running Time Analysis

Having described the algorithm of Theorem 2, we now discuss its running time. The process which computes the breakpoints in the proof of Lemma 1 runs in $O(|\mathcal{I}|)$ time. As described above, the solutions $\mathcal{I}_j^2$ can be stored while computing the breakpoints, and determining the solutions $\mathcal{I}_j^1$ requires just one scan of the subintervals. Therefore this step can also be implemented in $O(|\mathcal{I}|)$ time. Finally the construction of the solution $\mathcal{S}$ requires sorting the partial solutions for the blocks, which we can do in $O(\min\{n+|\mathcal{I}|, |\mathcal{I}|\log|\mathcal{I}|\})$ time. This step dominates the running time of the algorithm. This completes the proof of Theorem 2.

3 Proof of Theorem 1

In this section, we prove our main result. As in Section 2, we assume that our input has been preprocessed. We proceed analogously by decomposing the interval $I = [1, n]$ into blocks, however rather than requiring that each block can be covered by two subintervals of $\mathcal{I}$, we instead choose the blocks so that each one can be covered by a carefully chosen constant number of subintervals. This condition is formalized as follows.

Definition 7. An interval $J \subseteq I$ is ℓ-compact with respect to $\mathcal{I}$ if there exist ℓ subintervals $\{J_1, \ldots, J_\ell\} \subseteq \mathcal{I}|_J^P$ such that each point of J is covered by at least one of $J_1, \ldots, J_\ell$.

3.1 Partitioning the Interval

The key to the algorithm is once again the choice of the breakpoints. Let $c = \lceil (4\epsilon)^{-1} \rceil + 1$. We choose breakpoints $1 = b_0, b_1, ..., b_{k-1}$ which specify blocks $B_j = [b_{j-1}, b_j - 1]$ for $1 \leq j \leq k-1$ and $B_k = [b_{k-1}, n]$ (refer again to Figure 1). This time we choose the breakpoints such that they satisfy more sophisticated conditions.

Lemma 2. *There exists a choice of breakpoints $b_0, \ldots, b_{k-1}$ such that*

1. *every subinterval of $\mathcal{I}$ contains at most one breakpoint,*
2. *B_j is $16c$-compact for $1 \leq j \leq k$, and*
3. *for any $1 \leq j \leq k-1$, suppose $K \in \mathcal{I}$ is a subinterval which covers b_j. Then for some $j' \in \{j, j+1\}$, there exist at least $2c$ disjoint subintervals in $\mathcal{I}$ that are at least as long as K each of which covers only points in $B_{j'} \setminus \{b_{j'-1}\}$.*

Further, such breakpoints can be computed in $O(|\mathcal{I}|)$ time.

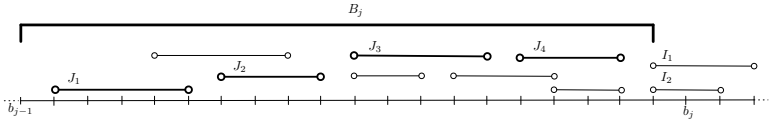


Fig. 3. An example illustrating Condition 3 of Lemma 2. Assuming $c = 1$, $B_j \setminus \{b_{j-1}\}$ contains at least 4 intervals, namely J_1, J_2, J_3 and J_4, each of which is at least as long as the two intervals, I_1 and I_2, which cover b_j.

For each ℓ between 1 and $16c$ and block B_j we determine a set of at most ℓ subintervals covering the most points in B_j. Then we find a $(1+\varepsilon)$-approximate solution to the original instance by greedily combining these partial solutions. The proof of Lemma 2 can be found in Section Appendix A. We now briefly example the three properties.

Properties 1 and 2 are analogous to their counterparts in Lemma 1, ensuring that each subinterval is contained in at most one block and that each block can be covered by a constant number of intervals. To understand Property 3, suppose that our choice of breakpoints were so fortuitous as to have the additional property that no subinterval simultaneously covered points in two different blocks. Then it is not difficult to see that this approach would yield an optimal solution. Intuitively speaking, we require Property 3 in Lemma 2 so that we force optimal solutions to prefer using subintervals that do not contain the breakpoints. This idea will be made precise in the analysis below.

3.2 Computing and Combining the Partial Solutions

Just as we did in the 2-approximation algorithm, we want to use greedy picking to select intervals from the blocks to form our solution. However now the blocks B_j are not 2-compact, but $16c$-compact, and so we need to keep more partial solutions.

Definition 8. *Let $1 \leq i \leq 16c$ and $1 \leq j \leq k$. We define $\mathcal{I}_j^i$ to be a collection of at most i subintervals from $\mathcal{I}|_{B_j}^P$ whose union covers the maximum number of points in B_j out of any such collection.*

An important observation is that for each block B_j, it is possible to find the solutions $\{\mathcal{I}_j^0, ..., \mathcal{I}_j^{16c}\}$ in $O(16c \cdot |\mathcal{I}|_{B_j}^P|)$ time.

Lemma 3. *There exists an algorithm which given a (preprocessed) collection $\mathcal{J}$ of subintervals of an interval J and a constant ℓ, finds $\mathcal{J}^0, ..., \mathcal{J}^\ell$ where for each $0 \leq i \leq \ell$, $\mathcal{J}^i$ is a subset of $\mathcal{J}$ such that $|\mathcal{J}^i| \leq i$ and the subintervals in $\mathcal{J}^i$ cover at least as many points as any other $\mathcal{K} \subseteq \mathcal{J}$ with $|\mathcal{K}| \leq i$, in $O(\ell \cdot |\mathcal{J}|)$ time.*

The proof of Lemma 3 uses dynamic programming and is omitted due to space considerations.

To determine our final solution $\mathcal{S}$, we use the greedy picking technique. The solution $\mathcal{S}$ will be a collection of preimages of between 0 and $16c$ subintervals from each block. Hence, a solution can be represented as a vector $x \in \{0, 1, ..., 16c\}^k$, where for each $j = 1, ..., k$, $\mathcal{I}_j^{x_j}$ is the set of intervals chosen from $\mathcal{I}|_{B_j}^P$. We find a vector $x^* \in \{0, 1, ..., 16c\}^k$ minimizing $cost(x)$ and subject to $\left|\bigcup_{j=1}^k \mathcal{I}_j^{x_j^*}\right| \geq rn$. We generalize Fact 6 to prove the following.

Definition 9. Let $x \in \{0, 1, ..., 16c\}^k$. For each $j \in 1, ..., k$ with $x_j < 16c$, define the *marginal of* $\mathcal{I}_j^{x_j+1}$ as $g(\mathcal{I}_j^{x_j+1}) = |\bigcup_{I \in \mathcal{I}_j^{x_j+1}} I| - |\bigcup_{I \in \mathcal{I}_j^{x_j}} I|$

Lemma 4. *For each $j = 1, ..., k$ and $i = 0, 1, ..., 16c$, $g(\mathcal{I}_j^i) \geq g(\mathcal{I}_j^{i+1})$.*

Since the marginal is always a nonnegative integer, Lemma 4 is equivalent to the statement that for each j the function $f_j(i) = \left|\bigcup_{J \in \mathcal{I}_j^i} J\right|$ is concave. For brevity we have omitted the proof of this fact.

Starting with $x^0 = (0, 0, ...0)$, we use the greedy picking technique: for each iteration $\ell = 1, 2, ...$, find x^ℓ by increasing an element of $x^{\ell-1}$ which gives the maximum possible marginal by 1. We stop when $\left|\bigcup_{j=1}^{k} \mathcal{I}_j^{x_j^\ell}\right| \geq rn$. A simple argument using Lemma 4 implies that at each iteration ℓ, x^ℓ covers as many points as any other solution whose cost is ℓ. Hence, our final solution is the desired vector x^*. The proof of this fact has been omitted.

3.3 Performance Analysis

We prove in this section that there exists a solution to PICP containing at most $(1+\epsilon)OPT$ subintervals. To do so, it is enough to show that there exists a vector $o \in \{0, 1, ..., 16c\}^k$ such that $cost(o) \leq (1+\epsilon)OPT$ and $\left|\bigcup_{j=1}^{k} \mathcal{I}_j^{o_j}\right| \geq rn$. Indeed, the existence of o implies the vector x^* found by the greedy picking technique also satisfies $cost(x^*) \leq (1+\epsilon)OPT$ and $|\bigcup_{j=1}^{k} \mathcal{I}_j^{x_j^*}| \geq rn$. We determine a solution of PICP corresponding to x^* by determining a collection $\mathcal{S}$ of preimages of the subintervals in $\bigcup_{j=1}^{k} \mathcal{I}_j^{x_j^*}$. It follows immediately that $\mathcal{S}$ is the desired solution.

To find such vector $o \in \{0, 1, ..., 16c\}^k$, we consider an optimal solution $\mathcal{I}^*$ to PICP in which the number of subintervals which cover a breakpoint in $\{b_1, \ldots, b_{k-1}\}$ is minimized over all optimal solutions. For each $1 \leq j \leq k$, define o_j^* as the number subintervals in $\mathcal{I}^*$ which only cover points in $B_j \setminus \{b_{j-1}\}$. For $0 \leq j \leq k-1$, define g_j^* as the number of subintervals in $\mathcal{I}^*$ which cover the breakpoint b_j. By optimality, $g_j^* \in \{0, 1, 2\}$. Since no subinterval contains two breakpoints, we have $OPT = \sum_{j=1}^{k} o_j^* + \sum_{j=0}^{k-1} g_j^*$, and note that $\mathcal{I}^*$ minimizes $\sum_{j=1}^{k-1} g_j^*$.

Now, let $o_1 = \min\{o_1^* + g_0^*, 16c\}$, for each $j \in \{2, \ldots, k-1\}$, $o_j = \min\{o_j^* + g_{j-1}^* + g_j^*, 16c\}$, and $o_k = \min\{o_k^* + g_{k-1}^*, 16c\}$.

Suppose that $g_j^* \geq 1$ for some $j \geq 1$. Then we claim that Lemma 2 and the minimality of $\mathcal{I}^*$ guarantee that either $o_j^* \geq c-1$ or $o_{j+1}^* \geq c-1$. To see this suppose to the contrary that there is some subinterval $K \in \mathcal{I}^*$ covering the breakpoint b_j, but $o_j^*, o_{j+1}^* < c-1$. Assume that there exists a set $\mathcal{J}$ of $2c$ disjoint subintervals, each of length at least $|K|$, covering only points in $B_j \setminus \{b_{j-1}\}$ (the case where the $2c$ subintervals cover points in B_{j+1} is similar). There are fewer than $c-1$ subintervals in $\mathcal{I}^*$ covering points in $B_j \setminus \{b_{j-1}\}$, and each of them can intersect at most two of the subintervals in $\mathcal{J}$, since no subinterval in $\mathcal{I}$ is contained in another. Further any subinterval in $\mathcal{I}^*$ covering b_{j-1} or b_j intersects at most one of the subintervals in $\mathcal{J}$. Therefore there is a subinterval $K' \in \mathcal{J}$, with $|K'| \geq |K|$ such that $\mathcal{I}^*$ does not cover any points in K'. This contradicts the minimality of $\mathcal{I}^*$, since $(\mathcal{I}^* \setminus K) \cup K'$ covers at least as many points in I as $\mathcal{I}^*$, while covering fewer breakpoints. Therefore, for each $1 \leq j \leq k-1$, we have $g_j^* \leq \frac{2}{c-1} o_j^*$ or $g_j^* \leq \frac{2}{c-1} o_{j+1}^*$, and so, $\sum_{j=1}^{k-1} g_j^* \leq 2 \sum_{j=1}^{k} \frac{2}{c-1} o_j^*$. Hence,

$$\sum_{j=1}^{k} o_j \leq \sum_{j=1}^{k} o_j^* + 2 \sum_{j=1}^{k-1} g_j^* \leq \sum_{j=1}^{k} o_j^* + \sum_{j=1}^{k-1} g_j^* + 2 \sum_{j=1}^{k} \frac{2}{c-1} o_j^* \leq OPT + \frac{4}{c-1} OPT = (1+\epsilon)OPT$$

Finally, we need to show that $\bigcup_{j=1}^{k} \mathcal{I}_j^{o_j}$ covers at least rn points. This follows easily because the optimal solution $\mathcal{I}^*$ uses at most o_j intervals from $\mathcal{I}|_{B_j}^P$. By definition of the o_j, the union of subintervals in $\mathcal{I}_j^{o_j}$ cover at least as many points in block B_j as the optimal solution $\mathcal{I}^*$ does.

3.4 Running Time Analysis

To finish the proof of Theorem 1, we need to analyze the running time. Lemma 2 ensures that the breakpoints can be computed in $O(|\mathcal{I}|)$ time. By Lemma 2, we see that for each j the computation of the solutions $\mathcal{I}_j^i$ takes time $O(16c \cdot |\mathcal{I}|_{B_j}^P|)$. As a consequence we can compute all of the $\mathcal{I}_j^i$ in time $O(16c \cdot \sum_j |\mathcal{I}|_{B_j}^P|) = O(\frac{|\mathcal{I}|}{\epsilon})$. Finally the construction of the solution $\mathcal{S}$ requires sorting the partial solutions for the blocks, which we can do in $O(\min\{n + |\mathcal{I}|, |\mathcal{I}| \log |\mathcal{I}|\})$ time. The running time of the algorithm is therefore $O(\frac{1}{\epsilon} \cdot \min\{n + |\mathcal{I}|, |\mathcal{I}| \log |\mathcal{I}|\})$.

This completes the proof of Theorem 1.

4 Concluding Remarks

We have presented two scalable algorithms for PICP: the linear time greedy 2-approximation algorithm and the near-linear $(1 + \varepsilon)$-approximation algorithm given by Theorem 1. The authors believe that the overarching paradigm discussed here of improving algorithms which are theoretically tractable to be scalable is a fundamental direction for future research. Natural generalizations of our work in this direction are to consider weighted PICP or even the *generalized partial set cover* studied by Könemann *et al.* in [18].

Acknowledgements. We would like to thank Howard Karloff for introducing us to this problem during a workshop at the Bellairs Institute, Barbados, in 2010. In addition we would like to thank Volodymyr Kuleshov for stimulating discussion during the workshop and Iraj Saniee for helpful comments on an earlier version of this manuscript.

References

1. Alon, N., Moshkovitz, D., Safra, S.: Algorithmic construction of sets for k-restrictions. ACM Transactions on Algorithms (TALG) 2(2), 153–177 (2006)
2. Bar-Yehuda, R.: Using homogeneous weights for approximating the partial cover problem. Journal of Algorithms 39(2), 137–144 (2001)
3. Bonchi, F., Castillo, C., Donato, D., Gionis, A.: Topical query decomposition. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, pp. 52–60. ACM, New York (2008)
4. Chvátal, V.: A Greedy Heuristic for the Set-Covering Problem. Mathematics of Operations Research 4(3), 233–235 (1979)
5. Cormen, T.H., Stein, C., Rivest, R.L., Leiserson, C.E.: Introduction to Algorithms, 2nd edn. McGraw-Hill Higher Education (2001)

6. Cormode, G., Karloff, H., Wirth, A.: Set cover algorithms for very large datasets. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM 2010, pp. 479–488. ACM, New York (2010)
7. Farahani, R.Z., Asgari, N., Heidari, N., Hosseininia, M., Goh, M.: Survey: Covering problems in facility location: A review. Comput. Ind. Eng. 62(1), 368–407 (2012)
8. Gandhi, R., Khuller, S., Srinivasan, A.: Approximation algorithms for partial covering problems. J. Algorithms 53(1), 55–84 (2004)
9. Gao, B.J., Ester, M., Cai, J.-Y., Schulte, O., Xiong, H.: The minimum consistent subset cover problem and its applications in data mining. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2007, pp. 310–319. ACM, New York (2007)
10. Golab, L., Karloff, H., Korn, F., Saha, A., Srivastava, D.: Sequential dependencies. Proc. VLDB Endow. 2(1), 574–585 (2009)
11. Golab, L., Karloff, H., Korn, F., Srivastava, D., Yu, B.: On generating near-optimal tableaux for conditional functional dependencies. Proc. VLDB Endow. 1(1), 376–390 (2008)
12. Gomes, F.C., Meneses, C.N., Pardalos, P.M., Viana, G.V.R.: Experimental analysis of approximation algorithms for the vertex cover and set covering problems. Comput. Oper. Res. 33(12), 3520–3534 (2006)
13. He, Z., Yang, C., Yu, W.: A Partial Set Covering Model for Protein Mixture Identification Using Mass Spectrometry Data. IEEE/ACM Trans. Comput. Biol. Bioinformatics 8(2), 368–380 (2011)
14. Johnson, D.S.: Approximation Algorithms for Combinatorial Problems. J. Comput. System Sci. 9, 256–278 (1974)
15. Karp, R.M.: Reducibility among combinatorial problems. In: 50 Years of Integer Programming 1958-2008, pp. 219–241 (2010)
16. Kearns, M.J.: The computational complexity of machine learning. The MIT Press (1990)
17. Kneis, J., Langer, A., Rossmanith, P.: Improved upper bounds for partial vertex cover. In: Broersma, H., Erlebach, T., Friedetzky, T., Paulusma, D. (eds.) WG 2008. LNCS, vol. 5344, pp. 240–251. Springer, Heidelberg (2008)
18. Könemann, J., Parekh, O., Segev, D.: A unified approach to approximating partial covering problems. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 468–479. Springer, Heidelberg (2006)
19. Lovász, L.: On the Ratio of Optimal Integral and Fractional Covers. Discrete Mathematics 13, 383–390 (1975)
20. Mihail, M.: Set Cover with Requirements and Costs Evolving over Time. In: Hochbaum, D.S., Jansen, K., Rolim, J.D.P., Sinclair, A. (eds.) RANDOM-APPROX 1999. LNCS, vol. 1671, pp. 63–72. Springer, Heidelberg (1999)
21. Raz, R., Safra, S.: A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, STOC 1997, pp. 475–484. ACM, New York (1997)
22. Slavík, P.: A tight analysis of the greedy algorithm for set cover. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, pp. 435–441. ACM (1996)

Appendix A Finding the Break Points

In this section, we prove Lemma 2. Recall that we are given an interval $I = [1, n]$, a collection of proper subintervals $\mathcal{I} = \{I_1, \ldots, I_t\}$ which are sorted by left endpoint, and a constant c. We describe an algorithm which determines breakpoints $b_1, \ldots, b_{k-1}$ such that 1) no subinterval contains two breakpoints, 2) the blocks determined by these breakpoints are each $16c$-compact, and 3) for each $1 \leq j \leq k-1$, for some $j' \in \{j, j+1\}$, there exist at least $2c$ disjoint subintervals in $\mathcal{I}$ that are at least as long as any subinterval covering b_j and each of which covers only points in $B_{j'} \setminus \{b_{j'-1}\}$.

A rough idea of the algorithm is as follows. We first scan through the subintervals in $\mathcal{I}$, marking some points $1 = d_1, d_2, \ldots, d_\ell, d_{\ell+1} = n+1$ as candidates to be breakpoints. For each candidate d_j $(1 \leq j \leq \ell)$, we will also keep track of a longest subinterval K_j in $\mathcal{I}$ which contains it. Crucially, our choice of candidates will ensure the following two properties hold:

i) For any $1 \leq i < j \leq \ell + 1$, the interval $[d_i, d_j - 1]$ is $(j-i)$-compact.
ii) The subintervals $K_2, K_4, K_6, \ldots$ are pairwise disjoint.
iii) No subinterval $I \in \mathcal{I}$ contains two or more distinct candidates.

We then choose the breakpoints from these candidates. Properties i), ii) and iii) will be the key to choosing the desired breakpoints in linear time.

Our algorithm proceeds to determine the breakpoints into two steps: The first, Algorithm 1, determines the candidates $d_1, d_2, \ldots, d_\ell$ and corresponding subintervals $K_1, K_2, \ldots, K_\ell$, and the second, Algorithm 2, uses these to determine the breakpoints.

Algorithm 1: Determining the Candidates

Input: $\mathcal{I} = \{I_1, I_2, \ldots, I_t\}$
Output: $\{K_1, K_2, \ldots, K_\ell\}$ and $\{d_1, d_2, \ldots, d_\ell\}$

```
1  begin
2      set covered = 0, i = 1, ℓ' = 1
3      while covered < n do
4          set d_ℓ' = covered + 1, K_ℓ' = ∅
5          while left(I_i) ≤ d_ℓ' do
6              if |I_i| > |K_ℓ'| then set K_ℓ' = I_i
7              set i = i + 1
8          end
           // J_ℓ' = I_{i-1}
9          set covered = right(I_{i-1}), ℓ' = ℓ' + 1
10     end
11 end
```

Algorithm 1 sets $d_1 = 1$ and K_1 to be the unique subinterval covering d_1. Given $K_1, ..., K_t$ and $d_1, ..., d_t$ for iteration $t \geq 1$, it sets d_{t+1} to be the left most uncovered point, namely $right(K_t) + 1$, and K_{t+1} to be the longest subinterval covering d_{t+1}. It scans through the subintervals exactly once, and hence, runs in $O(|\mathcal{I}|)$ time. We now prove that upon termination properties i), ii) and iii) hold. In doing so, we focus on the intervals $J_1, \ldots, J_\ell$, which we have included

in the algorithm to simplify its analysis. Note that the subintervals $J_1, \ldots, J_\ell$ cover the interval $I = [1, n]$. More strongly, defining $d_{\ell+1} = n + 1$, we have $[d_j, d_{j+1} - 1] \subseteq J_j$ for each $1 \leq j \leq \ell$, and so, for any $1 \leq i < j < \ell$, the interval $[d_i, d_{j+1} - 1]$ is $(j - i)$-compact, implying i) holds. In fact, J_j is the rightmost subinterval covering d_j, and so, $right(K_j) < right(J_j) + 1 = d_{j+1}$. From this fact it is straightforward to prove that ii) and iii) hold. Property ii) holds since J_{j+1} is the rightmost subinterval covering d_{j+1}, in particular, K_{j+2} does not cover d_{j+1}, hence, $left(K_{j+2}) > d_{j+1} > right(K_j)$. Property iii) holds since the existence any subinterval I containing both d_j and d_{j+1} would contradict that J_j is the rightmost subinterval containing d_i.

Algorithm 2 describes how given the subintervals $K_1, ..., K_\ell$ and candidates $d_1, d_2, ..., d_{\ell+1}$, we compute the breakpoints $b_0, b_1, ..., b_k$. The algorithm works with chunks of $8c$ candidate breakpoints and focuses on two sets, the $2c$ smallest even indexed candidates and the $2c$ largest even indexed candidates. We then choose the breakpoint to be d_t, one of these $4c$ candidates, such that its corresponding K_t is no longer than any of the subintervals corresponding to the remaining $4c - 1$ subintervals. We now prove that the runtime is $O(|\mathcal{I}|)$. For each j, the values of b_j^1 and b_j^2 can be computed by scanning through the intervals $K_i, K_{i+1}, ..., K_{i+8c-1}$ exactly once. Furthermore, we consider each of these subsets of $8c$ subintervals exactly once. Hence, Algorithm 2 takes $O(\ell) = O(|\mathcal{I}|)$ time. Hence the proof of Lemma 2 follows by showing that the determined breakpoints satisfy Properties 1, 2 and 3.

Algorithm 2: Determining the Break Points

Input: $\{K_1, K_2, \ldots, K_\ell\}$ and $\{d_1, d_2, \ldots, d_{\ell+1}\}$
Output: Break points $1 = b_0, b_1, \ldots, b_k = n + 1$ satisfying Properties 1,2, and 3

```
1  begin
2      set b_0 = 1, i = 1, j = 1
3      while i + 16c ≤ ℓ + 1 do
4          b_j^1 = argmin_{even t∈[i,i+4c)} |K_t|
5          b_j^2 = argmin_{even t∈[i+4c,i+8c)} |K_t|
6          if |K_{b_j^2}| ≤ |K_{b_j^1}| then set b_j = d_{b_j^2}
7          else set b_j = d_{b_j^1}
8          set i = i + 8c, j = j + 1
9      end
10     set b_j = n + 1
11 end
```

Property iii) of the candidates $d_1, ..., d_{\ell+1}$ immediately implies that Property 1 holds. For any two consecutive breakpoints b_j and b_{j+1} we have that there exists an i and indices ℓ' and ℓ'' such that $b_j = d_{\ell'}$, $d_{j+1} = d_{\ell''}$ and $i \leq \ell' < \ell'' \leq i+16c$. Hence, Property i) of Algorithm 1 implies $[b_j, b_{j+1} - 1]$ is $16c$-compact, satisfying Property 2. We need only show that Property 3 holds.

Consider any breakpoint b_j with $1 \leq j \leq k$. Without loss of generality assume that in this iteration $|K_{b_j^2}| \leq |K_{b_j^1}|$ (the argument when $|K_{b_j^2}| > |K_{b_j^1}|$ is symmetrical), and so $b_j = d_{b_j^2}$. Each of the $2c$ subintervals considered in computing

b_j^1 (i.e., the intervals with even indices of $K_i, K_{i+1}, ..., K_{i+4c-1}$) is at least as long as $K_{b_j^2}$ and disjoint by Property ii). We remind the reader that $K_{b_j^2}$ is a longest subinterval containing b_j. Furthermore, as i is odd at the beginning of each iteration, K_i is not one of these $2c$ intervals, and hence, b_{j-1} is not contained in any of these $2c$ interval $K_{i+1}, K_{i+3}, ..., K_{i+4c-1}$. Together these facts imply that $\mathcal{I}|_{B_j}^P$ contains at least $2c$ disjoint subintervals completely contained in $B_j - \{b_{j-1}\}$ which are at least as long as any subinterval containing b_j. Hence Property 3 holds, which completes the proof of Lemma 2.

Online Square-into-Square Packing

Sándor P. Fekete[1] and Hella-Franziska Hoffmann[2]

[1] Department of Computer Science, TU Braunschweig, Germany
s.fekete@tu-bs.de
[2] Cheriton School of Computer Science, University of Waterloo, Canada
hrhoffmann@uwaterloo.ca

Abstract. In 1967, Moon and Moser proved a tight bound on the critical density of squares in squares: any set of squares with a total area of at most 1/2 can be packed into a unit square, which is tight. The proof requires full knowledge of the set, as the algorithmic solution consists in sorting the objects by decreasing size, and packing them greedily into shelves. Since then, the online version of the problem has remained open; the best upper bound is still 1/2, while the currently best lower bound is 1/3, due to Han et al. (2008). In this paper, we present a new lower bound of 11/32, based on a dynamic shelf allocation scheme, which may be interesting in itself.

We also give results for the closely related problem in which the size of the square container is not fixed, but must be dynamically increased in order to accommodate online sequences of objects. For this variant, we establish an upper bound of 3/7 for the critical density, and a lower bound of 1/8. When aiming for accommodating an online sequence of squares, this corresponds to a 2.82...-competitive method for minimizing the required container size, and a lower bound of 1.33... for the achievable factor.

Keywords: Packing, online problems, packing squares, critical density.

1 Introduction

Packing is one of the most natural and common optimization problems. Given a set $\mathcal{O}$ of objects and a container E, find a placement of all objects into E, such that no two overlap. Packing problems are highly relevant in many practical applications, as well as in geometric and abstract settings. Simple one-dimensional variants (such as the PARTITION case with two containers, or the KNAPSACK problem of a largest packable subset) are NP-hard. Additional difficulties occur in higher dimensions: as Leung et al. [9] showed, it is NP-hard even to check whether a given set of squares fits into a unit-square container.

When dealing with an important, but difficult optimization problem, it is crucial to develop a wide array of efficient methods for distinguishing feasible instances from the infeasible ones. In one dimension, a trivial necessary and sufficient criterion is the total size of the objects in comparison to the container. This makes it natural to consider a similar approach for the two-dimensional

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 126–141, 2013.

version: *What is the largest number δ, such that any family of squares with area at most δ can be packed into a unit square?* An upper bound of $\delta \leq 1/2$ is trivial: two squares of size $1/2 + \varepsilon$ cannot be packed. As Moon and Moser showed in 1967 [11], $\delta = 1/2$ is the correct critical bound: sort the objects by decreasing size, and greedily pack them into a vertical stack of one-dimensional "shelves", i.e., horizontal subpackings whose height is defined by the largest object.

This approach cannot be used when the set of objects is not known a priori, i.e., in an online setting. It is not hard to see that a pure shelf-packing approach can be arbitrarily bad. However, other, more sophisticated approaches were able to prove lower bounds for δ: the current best bound (established by Han et al. [4]) is based on a relatively natural recursive approach and shows that $\delta \geq 1/3$.

Furthermore, it may not always be desirable (or possible) to assume a fixed container: the total area of objects may remain small, so a fixed large, square container may be wasteful. Thus, it is logical to consider the size of the container itself as an optimization parameter. Moreover, considering a possibly *larger* container reflects the natural optimization scenario in which the full set of objects *must* be accommodated, possibly by paying a price in the container size. From this perspective, $1/\sqrt{\delta}$ yields a competitive factor for the minimum size of the container, which is maintained at any stage of the process. This perspective has been studied extensively for the case of an infinite strip, but not for an adjustable square.

Our Results. We establish a new best lower bound of $\delta \geq 11/32$ for packing an online sequence of squares into a fixed square container, breaking through the threshold of $1/3$ that is natural for simple recursive approaches based on brick-like structures. Our result is based on a two-dimensional system of multi-directional shelves and buffers, which are dynamically allocated and updated. We believe that this approach is interesting by itself, as it may not only yield worst-case estimates, but also provide a possible avenue for further improvements, and be useful as an algorithmic method.

As a second set of results, we establish the first upper and lower bounds for a square container, which is dynamically enlarged, but must maintain its quadratic shape. In particular, we show that there is an upper bound of $\delta \leq 3/7 < 1/2$ for the critical density, and a lower bound of $1/8 \leq \delta$; when focusing on the minimum size of a square container, these results correspond to a $2.82\ldots$-competitive factor, and a lower bound of $1.33\ldots$ for the achievable factor by any deterministic online algorithm.

Related Work: Offline Packing of Squares. One of the earliest considered packing variants is the problem of finding a dense square packing for a rectangular container. In 1966 Moser [12] first stated the question as follows:

> "What is the smallest number A such that any family of objects with total area at most 1 can be packed into a rectangle of area A?"

The offline case has been widely studied since 1966; there is a long list of results for packing squares into a rectangle. Already in 1967, Moon and Moser [11] gave

the first bounds for A: any set of squares with total area at most 1 can be packed into a square with side lengths $\sqrt{2}$, which shows $A \leq 2$, and thus $\delta \geq 1/2$; they also proved $A \geq 1.2$. Meir and Moser [10] showed that any family of squares each with side lengths $\leq x$ and total area A can be packed into a rectangle of width w and height h, if $w, h \geq x$ and $x^2 + (w-x)(h-x) \geq A$; they also proved that any family of k-dimensional cubes with side lengths $\leq x$ and total volume V can be packed into a rectangular parallelepiped with edge lengths $a_1, \ldots, a_k$ if $a_i \geq x$ for $i = 1, \ldots, k$ and $x^k + \prod_{i=1}^{k} (a_i - x) \geq V$. Kleitman and Krieger improved the upper bound on A to $\sqrt{3} \approx 1.733$ [7] and to $4/\sqrt{6} \approx 1.633$ [8] by showing that any finite family of squares with total area 1 can be packed into a rectangle of size $\sqrt{2} \times 2/\sqrt{3}$. Novotný further improved the bounds to $1.244 \approx (2 + \sqrt{3})/3 \leq A < 1.53$ in 1995 [13] and 1996 [14]. The current best known upper bound of 1.3999 is due to Hougardy [5].

Online Packing of Squares. In 1997, Januszewski and Lassak [6] studied the online version of the dense packing problem. In particular, they proved that for $d \geq 5$, every online sequence of d-dimensional cubes of total volume $2(\frac{1}{2})^d$ can be packed into the unit cube. For lower dimensions, they established online methods for packing (hyper-) cubes and squares with a total volume of at most $\frac{3}{2}(\frac{1}{2})^d$ and $\frac{5}{16}$ for $d \in \{3, 4\}$ and $d = 2$, respectively. The results are achieved by performing an online algorithm that subsequently divides the unit square into rectangles with aspect ratio $\sqrt{2}$. In the following, we call these rectangles *bricks*. The best known lower bound of $2(\frac{1}{2})^d$ for any $d \geq 1$ was presented by Meir and Moser [10].

Using a variant of the brick algorithm, Han et al. [4] extended the result to packing a 2-dimensional sequence with total area $\leq 1/3$ into the unit square.

A different kind of online square packing was considered by Fekete et al. [2,3]. The container is an unbounded strip, into which objects enter from above in a Tetris-like fashion; any new object must come to rest on a previously placed object, and the path to its final destination must be collision-free. Their best competitive factor is $34/13 \approx 2.6154\ldots$, which corresponds to an (asymptotic) packing density of $13/34 \approx 0.38\ldots$.

2 Packing into a Fixed Container

As noted in the introduction, it is relatively easy to achieve a dense packing of squares in an offline setting: sorting the items by decreasing size makes sure that a shelf-packing approach places squares of similar size together, so the loss of density remains relatively small. This line of attack is not available in an online setting; indeed, it is not hard to see that a brute-force shelf-packing method can be arbitrarily bad if the sequence of items consists of a limited number of medium-sized squares, followed by a large number of small ones. Allocating different size classes to different horizontal shelves is not a remedy, as we may end up saving space for squares that never appear, and run out of space for smaller squares in the process; on the other hand, fragmenting the space for

large squares by placing small ones into it may be fatal when a large one does appear after all.

Previous approaches (in particular, the brick-packing algorithm) have sidestepped these difficulties by using a recursive subdivision scheme. While this leads to relatively good performance guarantees (such as the previous record of 1/3 for a competitive ratio), it seems impossible to tighten the lower bound; in particular, 1/3 seems to be a natural upper bound for this relatively direct approach. Thus, making progress on this natural and classical algorithmic problem requires less elegant, but more powerful tools.

In the following we present a different approach for overcoming the crucial impediment of mixed square sizes, and breaking through the barrier of 1/3. Our *Recursive Shelf Algorithm* aims at subdividing the set of squares into different size classes called *large*, *medium* and *small*, which are packed into pre-reserved shelves. The crucial challenge is to dynamically update regions when one of them gets filled up before the other ones do; in particular, we have to protect against the arrival of one large square, several medium-sized squares, or many small ones. To this end, we combine a number of new techniques:

- Initially, we assign carefully chosen horizontal strips for shelf-packing each size class.
- We provide rules for dynamically updating shelf space when required by the sequence of items. In particular, we accommodate a larger set of smaller squares by inserting additional *vertical* shelves into the space for larger squares whenever necessary.
- In order to achieve the desired overall density, we maintain a set of buffers for overflowing strips. These buffers can be used for different size classes, depending on the sequence of squares.

With the help of these techniques, and a careful analysis, we are able to establish $\delta \geq 11/32$. It should be noted that the development of this new technique may be more significant than the numerical improvement of the density bound: we are convinced that tightening the remaining gap towards the elusive 1/2 will be possible by an extended (but more complicated) case analysis.

In the following Section 2.1, we describe the general concept of shelf-packing. In Section 2.2 we give an overview of the algorithm. Section 2.3 sketches the placement of large objects, while Section 2.4 describes the packing created with medium-sized squares. The packing of small squares is discussed in Section 2.5. The overall performance is analyzed in Section 2.6. Due to limited space, various proof details and detailed pseudocode had to be omitted and can be found in the full version of the paper.

2.1 Shelf Packing

For a given subset of squares with maximum size h, a *shelf* $\mathcal{S}$ is a subrectangle of the container that has height h; a *shelf packing* places the squares into a shelf next to each other, until some object no longer fits; see Fig. 1(a). When that happens, the shelf is closed, and a new shelf gets opened.

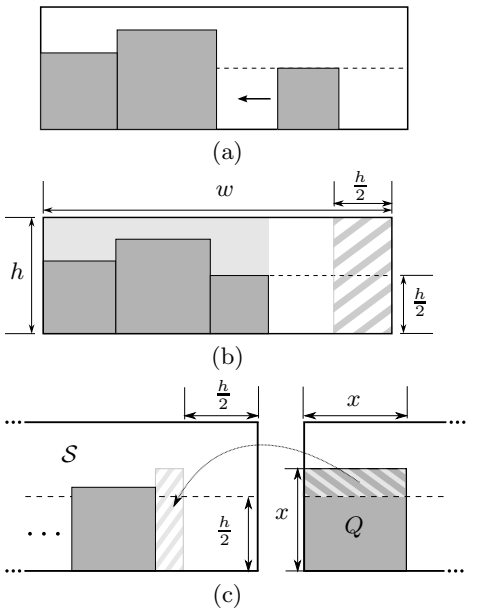


Fig. 1. (a) A shelf packed with squares of one height class. (b) Different areas of a shelf $\mathcal{S}$. *end*($\mathcal{S}$): hatched region to the right, *occupied*($\mathcal{S}$): total area of squares (dark gray) and *usedSection*($\mathcal{S}$): region with light gray background (incl. *occupied*($\mathcal{S}$)). (c) Assignment of the extra area of Q (hatched) to $\mathcal{S}$ when square Q causes on overflow of shelf $\mathcal{S}$.

In the following, we will subdivide the set of possible squares into subsets, according to their size: We let H_k denote the height class belonging to the interval $(2^{-(k+1)}, 2^{-k}]$. In particular, we call all squares in H_0 *large*, all squares in H_1 *medium*, and all other squares (in $H_{\geq 2}$) *small*.

The proof for the following useful lemma (according to Moon and Moser [11]) is straightforward and omitted.

Lemma 1. *Let $\mathcal{S}$ be a shelf of height class H_k with width w and height h that is packed with a set $\mathcal{P}$ of squares all belonging to H_k. Let Q be an additional square of H_k with side length x that does not fit into $\mathcal{S}$. Then the total area of all the squares packed into $\mathcal{S}$ plus the area of Q is greater than $\frac{\|\mathcal{S}\|}{2} - (h/2)^2 + x \cdot \frac{h}{2}$.*

Notation. In the following, we let $w_{\mathcal{S}}$ denote the width of a shelf $\mathcal{S}$, $h_{\mathcal{S}}$ denote its height and $\mathcal{P}(\mathcal{S})$ denote the set of squares packed into it. We define *usedSection*($\mathcal{S}$) as the horizontal section of $\mathcal{S}$ that contains $\mathcal{P}(\mathcal{S})$; see Fig. 1(b). We denote the (possibly empty) section with width $h_{\mathcal{S}}/2$ and height $h_{\mathcal{S}}$ at the end of $\mathcal{S}$ by *end*($\mathcal{S}$). The total area of the squares actually packed into a shelf $\mathcal{S}$ is *occupied*($\mathcal{S}$). The part of the square Q extending over the upper half of $\mathcal{S}$ is the *extra area* of Q. When Q causes a shelf $\mathcal{S}$ to be closed, we assign *extra*(Q) to $\mathcal{S}$; see Fig. 1(c). The total area assigned this way is referred to as *assigned*($\mathcal{S}$).

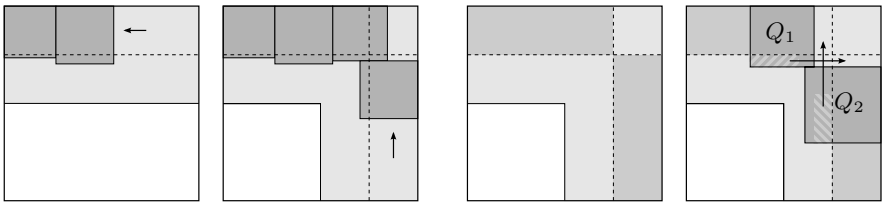


Fig. 2. (a): The L-shaped packing created with the squares of height class H_1. (b) Analysis of the Ceiling-Packing Algorithm: The algorithm packs at least as much as the gray area shown on the left. The parts of the packed squares that are used to fill the gaps appearing in the top right corner are depicted as hatched regions.

We let $\widetilde{\mathcal{A}}(\mathcal{S})$ denote the total of occupied and assigned area of $\mathcal{S}$ minus the extra area of the squares in $\mathcal{P}(\mathcal{S})$.

2.2 Algorithm Overview

We construct a shelf-based packing in the unit square by packing *small*, *medium* and *large squares* separately. We stop the Recursive Shelf Algorithm when the packings of two different subalgorithms would overlap. As it turns out, this can only happen when the total area of the given squares is greater than 11/32; details are provided in the "Combined Analysis" of Section 2.6, after describing the approach for individual size classes.

2.3 Packing Large Squares

The simplest packing subroutine is applied to large squares, i.e., of size greater than 1/2. We pack a square $Q_0 \in H_0$ into the top right corner of the unit square U. Clearly, only one large square can be part of a sequence with total area $\leq 11/32$. Hence, this single location for the squares in H_0 is sufficient.

2.4 Packing Medium Squares

We pack all medium squares (those with side lengths in $(1/4, 1/2]$) separately; note that there can be at most 5 of these squares, otherwise their total area is already bigger than $3/8 > 11/32$. Moreover, if there is a large square, there can be at most one medium square (otherwise the total area exceeds 3/8), and both can be packed next to each other.

We start with packing the H_1-squares from left to right coinciding with the top of the unit square U. If a square would cross the right boundary of U, we continue by placing the following squares from top to bottom coinciding with the right boundary; see Fig. 2(a).

We call the corresponding subroutine the *Ceiling Packing Algorithm*. Without interference of other height classes, the algorithm succeeds in packing any sequence of H_1-squares with total area $\leq 3/8$.

Theorem 2. *The Ceiling Packing Subroutine packs any sequence of medium squares with total area at most* $3/8$ *into the unit square.*

2.5 Packing Small Squares

As noted above, the presence of one large or few medium squares already assigns a majority of the required area, without causing too much fragmentation. Thus, the critical question is to deal with small squares in a way that leaves space for larger ones, but allows us to find extra space for a continuing sequence of small squares.

In the Recursive Shelf Algorithm we pack all small squares according to the *packSmall* subroutine, independent of the large and medium square packings. This method is also based on shelf packings. We partition the unit square into shelves for different height classes, such that for certain subsections we maintain a total packing density of $1/2$.

Distribution of the Shelves. The general partition of the unit square is depicted in Fig. 3(a). The regions $M_1, \ldots, M_4$ (in that order) act as shelves for height class H_2. We call the union M of the M_i the *main packing area*; this is the part of U that will definitely be packed with squares by our algorithm. The other regions may stay empty, depending on the sequence of incoming squares. The regions $B_1, \ldots, B_4$ provide shelves for H_3. We call the union B of the B_j the *buffer area*. In the region A we reserve H_k-shelf space for every $k \geq 4$. We call A the *initial buffer area*. The ends E_i of the main packing regions M_i serve as both: parts of the main packing region and additional buffer areas. We call the union E of the E_i the *end buffer area*.

Packing Approach. During the packing process, we maintain open shelves for all the height classes for which we already received at least one square as input; we pack them according to the shelf-packing scheme described above. We start the packing of small squares in the lower half $\mathcal{H}_\ell$ of U. The region M_1 serves as the first H_2-shelf. The left half of B_1 serves as the *initial buffer shelf* for H_3. As soon as we receive the first square of a height class H_k with $k \geq 4$, we open an *initial buffer shelf* with width $1/4$ and height 2^{-k} on top of the existing shelves in A. All of these shelves are packed from left to right; see Fig. 3(b) for the packing directions. In case the initial buffer for a height class H_k with ≥ 3 cannot accommodate another H_k-square, we continue packing H_k-squares into *vertical shelves* that we cut out of the main packing region. To achieve a packing density of $1/2$ for these vertical shelves, we assign and occupy additional *buffer space* in $B \cup E$.

The reason for choosing this unit-square subdivision and packing scheme is to establish the following lemma.

Lemma 3. *In each step of the algorithm, the total area of the small squares packed into* U *is at least* $\|usedSection(M) \setminus E\|/2$.

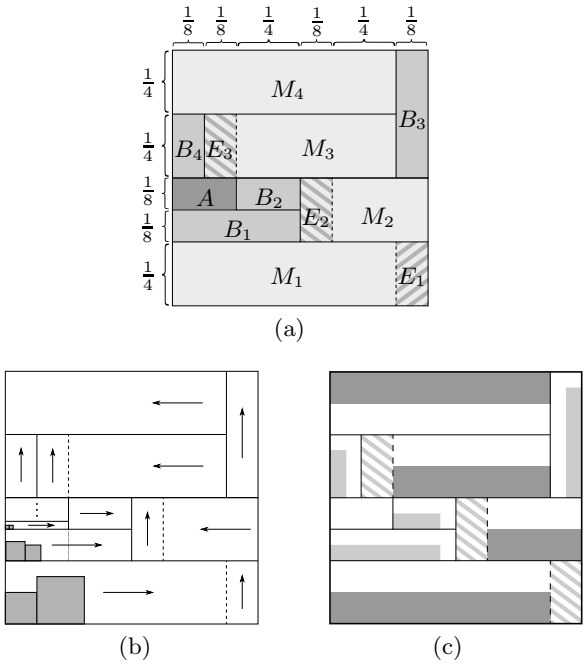


Fig. 3. (a) Region types in packingSmall. (b) Packing directions. (c) Area occupation in the main packing region (dark gray) and the usable buffer area (light gray).

We will argue that when the algorithm terminates, we have packed as much area as marked in Fig. 3(c). In the following, we describe the packings for the different small height classes in more detail.

Packing H_2-Squares. In the main packing area, we always maintain an open shelf for height class H_2 that is packed with H_2-squares, as described in Section 2.1. In order to avoid early collisions with large and medium squares, we start with packing M_1 from left to right, continuing with packing M_2 from right to left. Then we alternately treat M_3 and M_4 as the current main packing region, placing H_2-squares into the region whose *usedSection* is smaller. When the length of *usedSection*(M_4) becomes larger than 3/8, we prefer M_3 over M_4 until M_3 is full. This way, we can use the end E_3 as an additional buffer for vertical shelves in M_4 while ensuring that no overlap with the packing of H_1-squares can occur, unless the total area of the input exceeds 11/32. We know that each square of H_2 has a side length of at least half the height of the H_2-shelves. Consequently, if we only pack H_2-squares into the main packing area, the used sections of the M_i regions will be at least half full. However, if we receive a large number of $H_{\geq 3}$-squares as input, we may use these regions to accommodate $H_{\geq 3}$ as well.

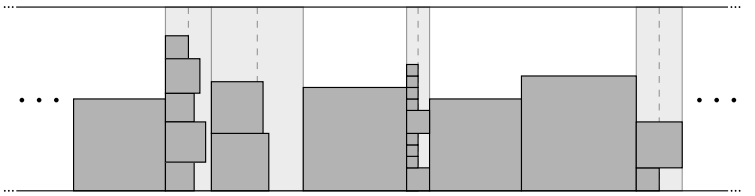


Fig. 4. Sample packing of square into a part of the main packing region according to the vertical shelf packing scheme

Packing of $H_{\geq 3}$-Squares. For $H_{k\geq 4}$ we reserve space in the initial buffer area A. As soon as we receive the first square of a height class H_k with $k \geq 4$, we open an H_k-shelf of length $1/4$ on top of the existing shelves in A. We call this shelf *initialBuffer*(k) and pack it from left to right with all subsequent H_k-squares according to the shelf-packing concept. If the *initialBuffer*(k) is full, we start allocating space in form of vertical H_k-shelves in the main packing region.

The same initial buffer packing is performed with H_3-squares, with the difference that we use B instead of A as the initial buffer packing location. As soon as the placement of a square Q would cause the initial buffer packing for H_3 in B to exceed a length of $1/4$, we open a vertical H_3-shelf with Q in the main packing region and stop using B for the initial buffer packing.

Vertical Shelves. In contrast to the situation in general shelf packings, we do not only pack H_2-squares into the H_2-shelves in the main packing region, but also cut out rectangular slices that serve as shelves for smaller height classes with a full initial buffer. We treat these vertical shelves as usual shelves for their corresponding height class. The width of each vertical main shelf for a height class H_k is always 2^{-k} and its height is always $1/4$, as it was formed by a slice of an H_2-shelf; see Fig. 4 for a sample packing. When closed, the resulting vertical shelf packings have a density of at least $3/8$; see Lemma 1. To achieve a total packing density of $1/2$ also for closed vertical shelves, we use the additional space reserved in the buffer area.

Buffer Assignments. There are three different ways to assign a buffer to a vertical shelf $\mathcal{S}$ for H_k with width $w_\mathcal{S}$.

1. If there is enough unassigned but occupied buffer area in region B, we simply assign a $w_\mathcal{S}/2$-wide slice of this area to $\mathcal{S}$.
2. Otherwise, we occupy further parts of the buffer area B by either
   (a) opening a $w_\mathcal{S}/2$-wide buffer subshelf $\mathcal{B}$ for height class H_k (if $k > 3$) or
   (b) simply packing the square into the buffer area (if $k = 3$).
   In both cases we assign a $(w_\mathcal{S}/2)^2$ part of the newly occupied area to $\mathcal{S}$.

If an overflow occurs at a vertical buffer subshelf for height class H_k, we want to pack the non-fitting square into the corresponding vertical main packing shelf $\mathcal{S}$. Hence, we start allocating buffer space for $\mathcal{S}$ when there is just enough space left in $\mathcal{S}$ to place another H_k-square. In particular, we open buffer subshelf $\mathcal{B}$ as soon as the next square would intersect *head*$(\mathcal{S})$, the top $w_\mathcal{S} \times w_\mathcal{S}$ part of $\mathcal{S}$.

In the following, we explain why this assignment is sufficient to establish the desired density in the main packing region.

By Lemma 1, we have $\widetilde{\mathcal{A}}(\mathcal{S}) \geq \|\mathcal{S} \setminus \mathit{end}(\mathcal{S})\|/2$ for any closed shelf $\mathcal{S}$ of our packing. Thus, for each vertical shelf $\mathcal{S}$ with width $w_{\mathcal{S}}$, we need to reserve an area of at most $\|\mathit{end}(\mathcal{S})\|/2 = (w_{\mathcal{S}}/2)^2$ in the buffer region in order to establish Lemma 3.

In case we create a vertical buffer subshelf $\mathcal{B}$ of width $w_{\mathcal{B}}$ $(=w_{\mathcal{S}})$ and height $h_{\mathcal{B}}$, we assign a $(w_{\mathcal{B}}/2)^2$-sized part of $\mathcal{B}$ to the corresponding vertical shelf $\mathcal{S}$ and have an area of $w_{\mathcal{B}}/2 \cdot h_{\mathcal{B}}/2$ to spare. We release this extra space as an occupied but unassigned buffer slice of density 1/2. We do the same with the extra area of an H_3-square that was placed in the buffer region. This way we ensure that every vertical shelf of width w effectively allocates a buffer of total length $w/2$ and that the area assigned in case 1 is sufficient. Thus, we get the following property of closed vertical main packing shelves; see the full paper for a detailed proof.

Lemma 4. *Let $\mathcal{S}$ be a closed vertical shelf for H_k that has a buffer part assigned to it. Then $\widetilde{\mathcal{A}}(\mathcal{S}) \geq \|\mathcal{S}\|/2$.*

The gaps remaining in an open vertical main packing shelf $\mathcal{S}$ may be larger. However, whenever we open a vertical shelf $\mathcal{S}$ for a height class H_k, we assign the occupied area of the initial buffer of H_k to it. This way we establish the following property.

Lemma 5. *If $\mathcal{S}$ is an open vertical shelf for H_k, then $\widetilde{\mathcal{A}}(\mathcal{S}) \geq \|\mathcal{S}\|/2$.*

Restricting the Used Buffer Region. To avoid early collisions with large and medium squares, we only pack new squares into the buffer region if there is not enough unassigned, occupied area remaining in the buffer regions.

Let $\mathcal{S}$ be a vertical main shelf for H_3, *bufferlength* be the total width of the occupied buffer sections and *requiredBufferLength* be the total buffer width required for the vertical shelves. If *bufferlength* $\geq$ *requiredBufferLength*, there must be an occupied but unassigned buffer slice, which we assign to $\mathcal{S}$. Otherwise, we need the area of an H_3-square Q to get the remaining buffer part. We distinguish two cases:

1. If *bufferlength*$+x \leq$ *requiredBufferLength*$+1/16$, we pack Q into B, ensuring the used buffer section to be at most 1/16 ahead of the section required.
2. Otherwise, the placement of Q in the buffer region would result in an undesirable buffer growth. In this case, we pack Q instead into its corresponding vertical main shelf and assign its potential extra space of $(x - 1/16) \cdot x$ to the buffer region. This way, a $(x - 1/16)$-wide buffer part is gained from Q, and the new buffer length is *bufferlength*$+ x - 1/16 >$ *requiredBufferLength*.

Buffer Region Overflow. A packing overflow may also occur in the buffer regions $B_1, \ldots, B_4$. In this case, we proceed as in every other shelf: we pack the current H_3-square or buffer subshelf into the next B_i-region in order and assign potential extra space from H_3-squares to the overflowing buffer shelf.

If an $H_{\geq 4}$-square cannot be placed at the end of *usedSection*(B_4), we pack it into the end of any of the other buffer regions of $B \cup E$. We claim that there must be enough space for this in at least one of the B_j-regions, as long as the total input area does not exceed 11/32.

By construction, we only need a $w/2$-wide buffer slice for each vertical main shelf with width w. Thus, the total buffer length required for $M \setminus E$ is at most 22/16. Unfortunately, the buffer region $B = B_1 \cup \cdots \cup B_4$ does only provide a total usable buffer length of 17/16 in the worst case. To get the missing buffer space of length 5/16, we also use the initial buffers and allocate extra space at the end of the main packing regions as buffer area.

Additional End Buffer Usage. The actual buffer space available in the E_i-regions depends on the lengths of the main packing sections that overlap with them. We call the part of *usedSection*(M_i) that overlaps with the considered E_i region $\mathcal{L}$. We denote the width of $\mathcal{L}$ by ℓ. Depending on ℓ, we choose between two different packing schemes for using E_i as a buffer.

1. *If* $\ell > w_{E_i}/2$*:* Then close the buffer shelf E_i (without having actively used it as an H_3-buffer-shelf) and simply use $\widetilde{\mathcal{A}}(\mathcal{L})$ as additional buffer area.
2. *If* $\ell \leq w_{E_i}/2$*:* Then treat $E_i \setminus \mathcal{L}$ similar to the normal buffer regions B_j; see Fig. 5(a).

We keep packing squares into $E_i \setminus \mathcal{L}$, according to the buffer-packing scheme, until the packing reaches a length of 2/16. We combine the area occupied this way with the area $\widetilde{\mathcal{A}}(\mathcal{L})$ of $\mathcal{L}$ to get proper buffers; see Fig. 5(b) and the full paper for pseudocode. We use 0.5/16 of the gained buffer length to achieve a density of 1/2 for $\mathcal{L}$ and hence get an additional buffer length of 1.5/16 from each of the E_i-regions. We start using (a part of) E_i as an additional buffer region, as soon as the corresponding main shelf M_i is closed. With the extra buffer length gained in E, we provide enough buffer space to fill the gaps remaining in all vertical main packing shelves in $M \setminus E$.

Lemma 6. *The total buffer space provided in* $A \cup B \cup E$ *is sufficient for the vertical shelves in* $M \setminus E$*.*

Lemmas 4, 5 and 6 directly prove the invariant of Lemma 3. That is, we can assume a density of 1/2 for the used sections of $M \setminus E$. By construction, the algorithm successfully packs all small squares, until a square Q would intersect the top left corner of U in M_4. At this time the total area of small input squares must be greater than $\|\bigcup_{i=1}^{3}(M_i \setminus E_i) \cup M_4\|/2 = 11/32$.

Theorem 7. *The packSmall Subroutine packs any sequence of small squares with total area at most* 11/32 *into the unit square.*

2.6 Combined Analysis

In the previous sections we proved that the algorithm successfully packs large, medium and large squares separately, as long as input has a total area of at most

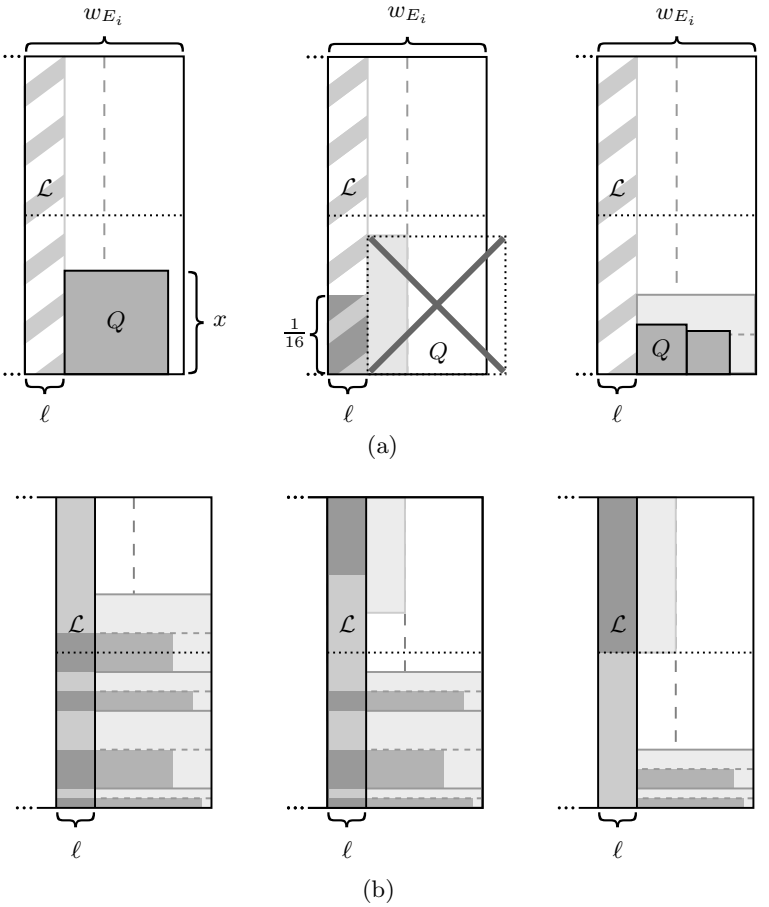


Fig. 5. (a) The packing performed in the end buffer regions: (left) packing of a fitting H_3-square, (center) extra area assignment from a non-fitting H_3-square and (right) subshelf packing of $H_{\geq 4}$ squares. (b) The assignment of $\widetilde{\mathcal{A}}(\mathcal{L})$, when E_i is closed: (left) we use all of $\widetilde{\mathcal{A}}(\mathcal{L})$ to extend the horizontal subshelves to a total length of w_{E_i}, (center) we use one half of $\widetilde{\mathcal{A}}(\mathcal{L})$ for the horizontal subshelf extensions and the other for an H_3-buffer or (right) we must use all of $\widetilde{\mathcal{A}}(\mathcal{L})$ for H_3-buffer assignment.

11/32. A case distinction over all possible collisions that may appear between the packings of these height classes can be used to prove the main result. An important property for this proof is the following; see the full paper for details.

Lemma 8. *Let Q be a small square with side length x in the buffer region B. Let ℓ be the total usable length of the buffer (value of bufferlength) right before Q was packed into B and let $\mathcal{P}$ be the set of small squares packed into U. Then the total area of the small input squares $\|\mathcal{P}\|$ is greater than $(\ell + x - 1/16) \cdot 1/4$.*

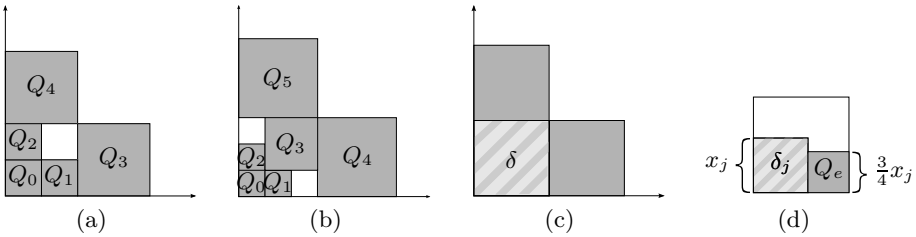


Fig. 6. Different choices in the lower-bound sequence: (a) Packing after choosing a side position. (b) Packing after choosing a center position. (c) The recurring pattern. (d) Packing a last square.

Theorem 9. *The Recursive Shelf Algorithm packs any sequence of squares with total area at most* 11/32 *into the unit square.*

3 Packing into a Dynamic Container

Now we discuss the problem of online packing a sequence of squares into a dynamic square container. At each stage, the container must be large enough to accommodate all objects; this requires keeping the container tight early on, but may require increasing its edge length appropriately during the process.

In the following, we give a non-trivial family of instances, which prove that no online algorithm can maintain a packing density greater than 3/7 for an arbitrary input sequence of squares and introduce an online square packing algorithm that maintains a packing density of 1/8 for an arbitrarily input sequence of squares.

3.1 An Upper Bound on δ

If the total area of the given sequence is unknown in advance, the problem of finding a dense online packing becomes harder. As it turns out, a density of 1/2 can no longer be achieved.

Theorem 10. *There are sequences for which no deterministic online packing algorithm can maintain a density strictly greater than* $3/7 \approx 0.4286$.

Proof. We construct an appropriate sequence of squares, depending on what choices a deterministic player makes; see Fig. 6. At each stage, the player must place a square Q_3 into a corner position (Fig. 6(a)) or into a center position (Fig. 6(b)); the opponent responds by either requesting another square of the same size (a), or two of the size of the current spanning box. This is repeated. A straightforward computation shows that the asymptotic density becomes 2/3, if the player keeps choosing side positions, and 3/4, if he keeps choosing center positions; for mixed choices, the density lies in between. Once the density is arbitrarily close to 3/4, with the center position occupied, the opponent can request a final square of size 3/4 of the current spanning box, for a density close

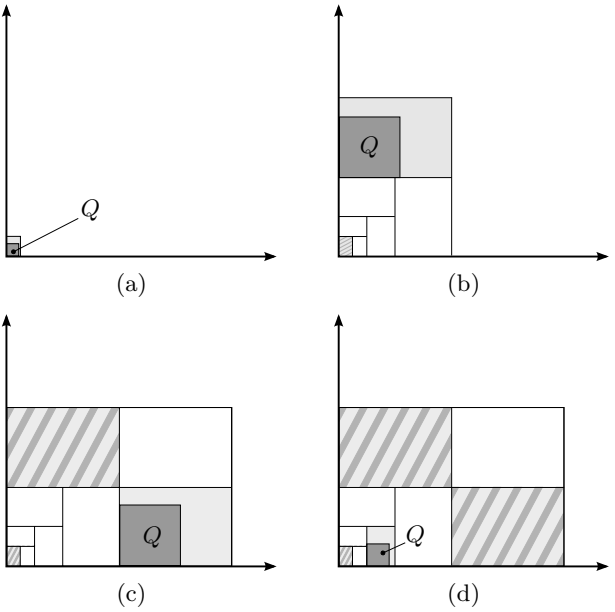


Fig. 7. The modified Brick-Packing algorithm for an input square Q. Occupied bricks are hatched, free bricks are blank. (a) A first square gets placed into the lower left corner, $B_{max} = S(Q)$. (b) If $S(Q) > B_{max}$, we double B_{max} until Q fits. (c) If Q does not fit into B_{max}, but $\|S(Q)\| < \|B_{max}\|$, we double B_{max} and subdivide the resulting brick. (d) If Q fits into B_{max}, we pack it into the smallest free fitting subbrick.

to $\frac{3/4+(3/4)^2}{(7/4)^2} = 3/7$; if the center position does not get occupied, the density is even worse. □

3.2 A Lower Bound on δ

When placing squares into a dynamic container, we cannot use our Recursive Shelf Algorithm, as it requires allocating shelves from all four container boundaries, which are not known in advance. However, we can adapt the Brick Algorithm by [6]: we consider bricks with side lengths equal to a power of $\sqrt{2}$ (and aspect ratio $1/\sqrt{2}$ or $\sqrt{2}$). We let B_k denote the brick of size $(\sqrt{2}^k, \sqrt{2}^{k+1})$ and let $S(Q)$ denote the smallest brick B_i that may contain a given square Q.

There are two crucial modifications: (1) The first square Q is packed into a brick of size $S(Q)$ with its lower left corner in the origin and (2) instead of always subdividing the existing bricks (starting with three fixed ones), we may repeatedly double the current maximum existing brick B_{max} to make room for large incoming squares. Apart from that, we keep the same packing scheme: Place each square Q into (a subbrick of) the smallest free brick that can contain Q; see Fig. 7 for an illustration.

Theorem 11. *For any input sequence of squares, the Dynamic Brick Algorithm maintains a packing density of at least* $1/8$.

Proof. By construction, every occupied brick has a density of at least $1/(2\sqrt{2})$. It is easy to see that in every step of the algorithm at most half the area of B_{max} consists of free bricks; compare [6]. Because B_{max} always contains all occupied bricks (and thus all packed squares), the ratio of $\|B_{max}\|$ to the area of the smallest enclosing square is at least $1/\sqrt{2}$. Therefore, the algorithm maintains an overall density of at least $(1/(2\sqrt{2})) \cdot (1/2) \cdot (1/\sqrt{2}) = 1/8$. □

3.3 Minimizing Container Size

The above results consider the worst-case ratio for the packing density. A closely related question is the online optimization problem of maintaining a square container with minimum edge length. The following is an easy consequence of Theorem 11, as a square of edge length $2\sqrt{2}$ can accommodate a unit area when packed with density $1/8$. By considering optimal offline packings for the class of examples constructed in Theorem 10, it is straightforward to get a lower bound of $4/3$ for any deterministic online algorithm.

Corollary 12. *Dynamic Brick Packing provides a competitive factor of* $2\sqrt{2} = 2.82\ldots$ *for packing an online sequence of squares into a square container with small edge length. The same problem has a lower bound of 4/3 for the competitive factor.*

4 Conclusion

We have presented progress on two natural variants of packing squares into a square in an online fashion. The most immediate open question remains the critical packing density for a fixed container, where the correct value may actually be less than $1/2$. Online packing into a dynamic container remains wide open. There is still slack in both bounds, but probably more in the lower bound.

There are many interesting related questions. What is the critical density (offline and online) for packing circles into a unit square? This was raised by Demaine et al. [1]. In an offline setting, there is a lower bound of $\pi/8 = 0.392\ldots$, and an upper bound of $\frac{2\pi}{(2+\sqrt{2})^2} = 0.539\ldots$, which is conjectured to be tight. Another question is to consider the critical density as a function of the size of the largest object. In an offline context, the proof by Moon and Moser provides an answer, but little is known in an online setting.

References

1. Demaine, E.D., Fekete, S.P., Lang, R.J.: Circle packing for origami design is hard. In: Origami5, pp. 609–626. AK Peters/CRC Press (2011)
2. Fekete, S.P., Kamphans, T., Schweer, N.: Online square packing. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tóth, C.D. (eds.) WADS 2009. LNCS, vol. 5664, pp. 302–314. Springer, Heidelberg (2009)
3. Fekete, S.P., Kamphans, T., Schweer, N.: Online square packing with gravity. Algorithmica (to appear, 2013)
4. Han, X., Iwama, K., Zhang, G.: Online removable square packing. Theory of Computing Systems 43(1), 38–55 (2008)
5. Hougardy, S.: On packing squares into a rectangle. Computational Geometry: Theory and Applications 44(8), 456–463 (2011)
6. Januszewski, J., Lassak, M.: On-line packing sequences of cubes in the unit cube. Geometriae Dedicata 67(3), 285–293 (1997)
7. Kleitman, D., Krieger, M.: Packing squares in rectangles I. Annals of the New York Academy of Sciences 175, 253–262 (1970)
8. Kleitman, D.J., Krieger, M.M.: An optimal bound for two dimensional bin packing. In: 16th Annual Symposium on Foundations of Computer Science (FOCS), pp. 163–168 (1975)
9. Leung, J.Y.-T., Tam, T.W., Wong, C.S., Young, G.H., Chin, F.Y.L.: Packing squares into a square. J. Parallel and Dist. Comp. 10(3), 271–275 (1990)
10. Meir, A., Moser, L.: On packing of squares and cubes. Journal of Combinatorial Theory 5(2), 126–134 (1968)
11. Moon, J., Moser, L.: Some packing and covering theorems. Colloquium Mathematicum 17, 103–110 (1967)
12. Moser, L.: Poorly formulated unsolved problems of combinatorial geometry. Mimeographed (1966)
13. Novotný, P.: A note on a packing of squares. Stud. Univ. Transp. Commun. Žilina Math.-Phys. Ser. 10, 35–39 (1995)
14. Novotný, P.: On packing of squares into a rectangle. Arch. Math. (Brno) 32(2), 75–83 (1996)

Online Non-clairvoyant Scheduling to Simultaneously Minimize All Convex Functions

Kyle Fox[1,⋆], Sungjin Im[2], Janardhan Kulkarni[2,⋆⋆], and Benjamin Moseley[3]

[1] Department of Computer Science, University of Illinois, Urbana, IL 61801
kylefox2@illinois.edu
[2] Department of Computer Science, Duke University, Durham NC 27708
{sungjin,kulkarni}@cs.duke.edu
[3] Toyota Technological Institute at Chicago, Chicago, IL 60637
moseley@ttic.edu

Abstract. We consider scheduling jobs online to minimize the objective $\sum_{i\in[n]} w_i g(C_i - r_i)$, where w_i is the weight of job i, r_i is its release time, C_i is its completion time and g is any non-decreasing convex function. Previously, it was known that the clairvoyant algorithm Highest-Density-First (HDF) is $(2+\epsilon)$-speed $O(1)$-competitive for this objective on a single machine for any fixed $0 < \epsilon < 1$ [1]. We show the first non-trivial results for this problem when g is not concave and the algorithm must be *non-clairvoyant*. More specifically, our results include:

- A $(2+\epsilon)$-speed $O(1)$-competitive non-clairovyant algorithm on a single machine for all non-decreasing convex g, matching the performance of HDF for any fixed $0 < \epsilon < 1$.
- A $(3+\epsilon)$-speed $O(1)$-competitive non-clairovyant algorithm on multiple identical machines for all non-decreasing convex g for any fixed $0 < \epsilon < 1$.

Our positive result on multiple machines is the first non-trivial one even when the algorithm is clairvoyant. Interestingly, all performance guarantees above hold for all non-decreasing convex functions g *simultaneously.* We supplement our positive results by showing any algorithm that is oblivious to g is not $O(1)$-competitive with speed less than 2 on a single machine. Further, any non-clairvoyent algorithm that knows the function g cannot be $O(1)$-competitive with speed less than $\sqrt{2}$ on a single machine or speed less than $2 - \frac{1}{m}$ on m identical machines.

1 Introduction

Scheduling a set of jobs that arrive over time on a single machine is perhaps the most basic setting considered in scheduling theory. A considerable amount

⋆ Research by this author is supported in part by the Department of Energy Office of Science Graduate Fellowship Program (DOE SCGF), made possible in part by the American Recovery and Reinvestment Act of 2009, administered by ORISE-ORAU under contract no. DE-AC05-06OR23100.

⋆⋆ Supported by NSF awards CCF-1008065 and IIS-0964560.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 142–157, 2013.

of work has focused on this fundamental problem. For examples, see [2]. In this setting, there are n jobs that arrive over time, and each job i requires some processing time p_i to be completed on the machine. In the *online* setting, the scheduler becomes first aware of job i at time r_i when job i is released. Note that in the online setting, it is standard to assume jobs can be *preempted*.

Generally, a client that submits a job i would like to minimize the *flow time* of the job defined as $F_i := C_i - r_i$, where C_i denotes the completion time of job i. The flow time of a job measures the amount of time the job waits to be satisfied in the system. When there are multiple jobs competing for service, the scheduler needs to make scheduling decisions to optimize a certain global objective. One of the most popular objectives is to minimize the total (or equivalently average) flow time of all the jobs, i.e., $\sum_{i\in[n]} F_i$. It is well known that the algorithm Shortest-Remaining-Processing-Time (SRPT) is optimal for that objective in the single machine setting. The algorithm SRPT always schedules the job that has the shortest remaining processing time at each point in time. Another well known result is that the algorithm First-In-First-Out (FIFO) is optimal for minimizing the maximum flow time, i.e., $\max_{i\in[n]} F_i$ on a single machine. The algorithm FIFO schedules the jobs in the order they arrive.

These classic results have been extended to the case where jobs have priorities. In this extension, each job i is associated with a weight w_i denoting its priority; large weight implies higher priority. The generalization of the total flow time problem is to minimize the total weighted flow time, $\sum_{i\in[n]} w_i F_i$. For this problem it is known that no online algorithm can be $O(1)$-competitive [3]. A generalization of the maximum flow time problem is to minimize the maximum weighted flow time $\max_{i\in[n]} w_i F_i$. It is also known for this problem that no online algorithm can be $O(1)$-competitive [4,5].

Due to these strong lower bounds, previous work for these objectives has appealed to the relaxed analysis model called resource augmentation [6]. In this relaxation, an algorithm A is said to be s-speed c-competitive if A has a competitive ratio of c when processing jobs s times faster than the adversary. The primary goal of a resource augmentation analysis is to find the minimum speed an algorithm requires to be $O(1)$-competitive. For the total weighted flow time objective, it is known that the algorithm Highest-Density-First (HDF) is $(1+\epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive for any fixed $\epsilon > 0$ [7,8]. The algorithm HDF always schedules the job i of highest density, $\frac{w_i}{p_i}$. For the maximum weighted flow objective, the algorithm Biggest-Weight-First (BFW) is known to be $(1+\epsilon)$-speed $O(\frac{1}{\epsilon})$-competitive [5]. BFW always schedules the job with the largest weight.

Another widely considered objective is minimizing the ℓ_k-norms of flow time, $\left(\sum_{i\in[n]} F_i^k\right)^{1/k}$ [9,10,11,12,13,14]. The ℓ_k-norm objective is most useful for $k \in \{1, 2, 3, \infty\}$. Observe that total flow time is the ℓ_1-norm of flow time, and the maximum flow time is the ℓ_∞-norm. The ℓ_2 and ℓ_3 norms are natural balances between the ℓ_1 and ℓ_∞ norms. These objectives can be used to decrease the variance of flow time, thereby yielding a schedule that is fair to requests. It is known that no algorithm can be $n^{\Omega(1)}$-competitive for minimizing the ℓ_2-norm

[9]. On the positive side, for $\epsilon > 0$, HDF was shown to be $(1+\epsilon)$-speed $O(\frac{1}{\epsilon^2})$-competitive for any ℓ_k-norm objective, $k \geq 1$ [9].

These objectives have also been considered in the identical machine scheduling setting [15,16,17,18,19,20,21,22]. In this setting, there are m machines that the jobs can be scheduled on. Each job can be scheduled on any machine and job i requires processing time p_i no matter which machine it is assigned to. In the identical machine setting it is known that any randomized online algorithm has competitive ratio $\Omega(\min\{\frac{n}{m}, \log P\})$, where P denotes the ratio between the maximum and minimum processing time of a job [15]. HDF as well as several other algorithms are known to be scalable for weighted flow time [8,16,22,21]. For the ℓ_k-norms objective the multiple machine version of HDF is known to be scalable [21] as well as other algorithms [16,22]. For the maximum unweighted flow it is known that FIFO is $(3-2/m)$-competitive, and for weighted maximum flow time a scalable algorithm is known [4,5].

The algorithms HDF and SRPT use the processing time of a job to make scheduling decisions. An algorithm which learns the processing time of a job upon its arrival is called *clairvoyant*. An algorithm that does not know the processing time of a job before completing the job is said to be *non-clairvoyant*. Among the aforementioned algorithms, FIFO and BFW are non-clairvoyant. Non-clairvoyant schedulers are highly desirable in many real world settings. For example, an operating system typically does not know a job's processing time. Thus, there has been extensive work done on designing non-clairvoyant schedulers for the problems discussed above. Scalable non-clairvoyant algorithms are known for the maximum weighted flow time, average weighted flow time, and ℓ_k-norms of flow time objectives even on identical machines [5,16].

It is common in scheduling theory that algorithms are tailored for specific scheduling settings and objective functions. For instance, FIFO is considered the best algorithm for non-clairvoyantly minimizing the maximum flow time, while HDF is considered one of the best algorithms for minimizing total weighted flow time. One natural question that arises is what to do if a system designer wants to minimize several objective functions simultaneously. For instance, a system designer may want to optimize average quality of service, while minimizing the maximum waiting time of a job. Different algorithms have been considered for minimizing average flow time and maximum flow time, but the system designer would like to have a single algorithm that performs well for both objectives.

Motivated by this question, the general cost function objective was considered in [1]. In the general cost function problem, there is a function $g : \mathbb{R}^+ \to \mathbb{R}^+$ given, and the goal of the scheduler is to minimize $\sum_{i\in[n]} w_i g(F_i)$. One can think of $g(F_i)$ as the penalty of making job i wait F_i time steps, scaled by job i's priority (its weight w_i). This objective captures most scheduling metrics. For example, this objective function captures total weighted flow time by setting $g(x) = x$. By setting $g(x) = x^k$, the objective also captures minimizing $\sum_{i\in[n]} F_i^k$ which is essentially the same as the ℓ_k-norm objective except the outer kth root is not taken. Finally, by making g grow very quickly the objective can be designed to capture minimizing the maximum weighted flow time. As stated, one of the

reasons this objective was introduced was to find an algorithm that can optimize several objectives simultaneously. If one were to design an algorithm that optimizes the general cost function g while being oblivious to g, then this algorithm would optimize *all* objective functions in this framework *simultaneously*.

In [1], the general cost function objective was considered only assuming that g is non-decreasing. This is a natural assumption since there should be no incentive for a job to wait longer. It was shown that in this case, no algorithm that is oblivious to the cost function g can be $O(1)$-competitive with speed $2-\epsilon$ for any fixed $\epsilon > 0$. Surprisingly, it was also shown that HDF, an algorithm that is oblivious to g, is $(2+\epsilon)$-speed $O(1/\epsilon)$-competitive. This result shows that it is indeed possible to design an algorithm that optimizes most of the reasonable scheduling objectives simultaneously on a single machine. Recall that HDF is clairvoyant. Ideally, we would like to have a non-clairvoyant algorithm for general cost functions. Further, there is currently no known similar result in the multiple identical machines setting.

Results: In this paper, we consider non-clairvoyant online scheduling to minimize the general cost function on a single machine as well as on multiple identical machines. In both the settings, we give the *first* nontrivial positive results when the online scheduler is required to be non-clairvoyant. We concentrate on cost functions g which are differentiable, non-decreasing, and convex. We assume without loss of generality that $g(0) = 0$. Note that all of the objectives discussed previously have these properties. We show the following somewhat surprising result (Section 4).

Theorem 1. *There exists a non-clairvoyant algorithm that is* $(2+\epsilon)$*-speed* $O(1/\epsilon)$*-competitive for minimizing* $\sum_{i\in[n]} w_i g(C_i - r_i)$ *on a single machine for any* $\epsilon > 0$*, when the given cost function* $g : \mathbb{R}^+ \to \mathbb{R}^+$ *is differentiable, non-decreasing, and convex (*g' *is non-decreasing). Further, this algorithm is oblivious to* g*.*

We then consider the general cost function objective on multiple machines for the first time, and give a positive result. This algorithm is also non-clairvoyant.

Theorem 2. *There exists a non-clairvoyant algorithm that is* $(3+\epsilon)$*-speed* $O(1/\epsilon)$*-competitive for minimizing* $\sum_{i\in[n]} w_i g(C_i - r_i)$ *on multiple identical machines for any* $\epsilon > 0$*, when the given cost function* $g : \mathbb{R}^+ \to \mathbb{R}^+$ *is differentiable, non-decreasing, and convex (*g' *is non-decreasing). Further, this algorithm is oblivious to* g*.*

Note that we do not know if there exists a constant competitive non-clairvoyant algorithm even for a single machine with any constant speed when the cost function is neither convex nor concave. We leave this gap as an open problem.

We complement these positive results by extending the lower bound presented in [1]. They showed that for any $\epsilon > 0$, no oblivious algorithm can be $(2-\epsilon)$-speed $O(1)$-competitive on a single machine when the cost function g is non-decreasing, but perhaps discontinuous. We show the same lower bound even if g is differentiable, non-decreasing, and convex. Thus, on a single machine, our positive result is essentially tight up to constant factors in the competitive ratio, and our algorithm achieves the same performance guarantee while being non-clairvoyant.

Theorem 3. *No randomized clairvoyant algorithm that is oblivious to g can be $(2-\epsilon)$-speed $O(1)$-competitive for minimizing $\sum_{i\in[n]} w_i g(C_i - r_i)$ on a single machine even if all jobs have unit weights and g is differentiable, non-decreasing, and convex.*

We go on to show that even if a non-clairvoyant algorithm knows the cost function g, the algorithm cannot have a bounded competitive ratio when given speed less than $\sqrt{2}$.

Theorem 4. *Any deterministic non-clairvoyant (possibly aware of g) algorithm for minimizing $\sum_{i\in[n]} w_i g(C_i - r_i)$ on a single machine has an unbounded competitive ratio when given speed $\sqrt{2}-\epsilon$ for any fixed $\epsilon > 0$ where g is differentiable, non-decreasing, and convex..*

Finally, we show that at least $2 - \frac{1}{m}$ speed is needed for any non-clairvoyant algorithm to be constant competitive on m identical machines. This is the first lower bound for the general cost function specifically designed for the multiple machine case.

Theorem 5. *Any randomized non-clairvoyant (possibly aware of g) algorithm on m identical machines has an unbounded competitive ratio when given speed less than $2 - \frac{1}{m} - \epsilon$ for any fixed $\epsilon > 0$ when g is differentiable, non-decreasing, and convex..*

Techniques: To show Theorem 1, we consider the well-known algorithm Weighted-Shortest-Elapsed-Time-First (WSETF) on a singe machine, and first show that it is 2-speed $O(1)$-competitive for minimizing the *fractional* version of the general cost function objective. Then with a small extra amount of speed augmentation, we convert WSETF's schedule into the one that is $(2+\epsilon)$-speed $O(1)$-competitive for the integral general cost function. This conversion is now a fairly standard technique, and will be further discussed in Section 2. This conversion was also used in [1] when analyzing HDF. One can think of the fractional objective as converting each job i to a set of p_i unit sized jobs of weight w_i/p_i. That is, the weight of the job is distributed among all unit pieces of the job. Notice that the resulting weight of the unit time jobs as well as the number of them depends on the job's original processing time. Thus, to analyze a non-clairvoyant algorithm for the fractional instance one must consider the algorithm's decisions on the original instance and argue about the algorithm's cost on the fractional instance. This differs from the analysis of [1], where the clairvoyant algorithm HDF can assume full knowledge of the conversion. Due to this, in [1] they can argue directly about HDF's decisions for the fractional instance of the problem. Since a non-clairvoyant algorithm does not know the fractional instance, it seems difficult to adapt the techniques of [1] when analyzing a non-clairvoyant algorithm.

If the instance consists of a set of unweighted jobs, WSETF always processes the job which has been processed the least. Let $q_i^A(t)$ be the amount WSETF has processed job i by time t. When jobs have weights, WSETF processes the job i such that $\frac{w_i}{q_i^A(t)}$ is maximized where w_i is the weight in the integral

instance. One can see that WSETF will not necessarily process the jobs with the highest weight at each time, which is what the algorithm HDF will do if all jobs are unit sized. Further, WSETF may round robin among multiple jobs of the same priority. For these reasons, our analysis of WSETF is substantially different from the analysis in [1], and relies crucially on a new lower bound we develop on the optimal solution. This lower bound holds for any objective that is differentiable, non-decreasing, and convex. Our lower bound gives a way to relate the final objective of the optimal solution to the volume of unsatisfied work the optimal solution has at each moment in time. We then bound the volume of unsatisfied jobs in the optimal schedule at each moment in time and relate this to WSETF's instantaneous increase in its objective function. We believe that our new lower bound will be useful in further analysis of scheduling problems since it is versatile enough to be used for many scheduling objectives.

Other Related Work: For minimizing average flow time on a single machine, the non-clairvoyant algorithms Shortest Elapse Time First (SETF) and Latest Arrival Processor Sharing (LAPS) are known to be scalable [6,23]. Their weighted versions Weighted Shortest Elapse Time First (WSETF) and Weighted Latest Arrival Processor Sharing (WLAPS) are scalable for average weighted flow time [9,24], and also for (weighted) ℓ_k norms of flow time [9,10].

In [1], Im et al. showed Weighted Latest Arrival Processor Sharing (WLAPS) is scalable for concave functions g. They also showed that no online randomized algorithm, even with any constant speed-up, can have a constant competitive ratio, when each job i has its own cost function g_i, and the goal is to minimize $\sum_{i\in[n]} g_i(F_i)$. This more general problem was studied in the offline setting by Bansal and Pruhs [25]. They gave an $O(\log\log nP)$-approximation (without speed augmentation), where P is the ratio of the maximum to minimum processing time of a job. This is the best known approximation for minimizing average weighted flow time offline, and a central open question in scheduling theory is whether or not a $O(1)$-approximation exists for weighted flow time offline.

2 Preliminaries

The Fractional Objective: In this section we define the fractional general cost objective and introduce some notation. We will refer to the non-fractional general cost objective as *integral*. For a schedule, let $p_i(t)$ denote the remaining processing time of job i at time t. Let $\beta_i(p)$ be the latest time t such that $p_i(t) = p$ for any p where $0 \leq p \leq p_i$.

The fractional objective penalizes jobs over time by charging in proportion to how much of the job remains to be processed. Formally, the fractional objective is defined as:

$$\sum_{i\in[n]} \int_{t=r_i}^{C_i} \frac{w_i p_i(t)}{p_i} g'(t-r_i)\mathrm{dt} \tag{1}$$

Generally when the fractional objective is considered, it is stated in the form (1). For our analysis it will be useful to note that this objective is equivalent to:

$$\sum_{i\in[n]} \frac{w_i}{p_i} \int_{p=0}^{p_i} g(\beta_i(p) - r_i)\mathrm{dp} \tag{2}$$

As noted earlier, considering the fractional objective has proven to be quite useful for the analysis of algorithms in scheduling theory, because directly arguing about the fractional objective is usually easier from an analysis viewpoint. A schedule which optimizes the fractional objective can then be used to get a good schedule for the integral objective as seen in the following theorems. In the first theorem (6), the algorithm's fractional cost is compared against the optimal solution for the fractional objective. In the second theorem (7), the algorithm's fractional cost is compared against the optimal solution for the integral instance.

Theorem 6 ([1]). *If a (non-clairvoyant) algorithm A is s-speed c-competitive for minimizing the fractional general cost function then there exists a $(1+\epsilon)s$-speed $\frac{(1+\epsilon)c}{\epsilon}$-competitive (non-clairvoyant) algorithm for the integral general cost function objective for any $0 \leq \epsilon \leq 1$.*

Theorem 7 ([1]). *If a (non-clairvoyant) algorithm A with s-speed has fractional cost at most a factor c larger than the optimal solution for the integral objective then there exists a $(1+\epsilon)s$-speed $\frac{(1+\epsilon)c}{\epsilon}$-competitive (non-clairvoyant) algorithm for the integral general cost function objective for any $0 \leq \epsilon \leq 1$.*

These two theorems follow easily by the analysis given in [1]. We note that the resulting algorithm that performs well for the integral objective is not necessarily the algorithm A. Interestingly, [1] shows that if A is HDF then the resulting algorithm is still HDF. However, if A is WSETF, the resulting integral algorithm need not be WSETF.

Notation: We now introduce some more notation that will be used throughout the paper. For a schedule B, let C_i^B be the completion time of job i. Let $p_i^B(t)$ denote the remaining processing time for job i at time t. Let $q_i^B(t) = p_i - p_i^B(t)$ be the amount job i has been processed by time t. Let $p_{i,j}^B(t) = (\min\{\frac{w_j}{w_i}p_i, p_j\} - q_j^B(t))^+$. Here $(\cdot)^+$ denotes $\max\{\cdot, 0\}$. Let $p_{i,j} = \min\{\frac{w_j}{w_i}p_i, p_j\} = p_{i,j}^B(r_j)$. If the schedule B is that produced by WSETF and $t \in [r_i, C_i^B]$ then $p_{i,j}^B(t)$ is exactly the amount of processing time WSETF will devote to job j during the interval $[t, C_i^B]$. In other words, the remaining time job i waits due to WSETF processing job j. Let $Q_B(t)$ be the set of job released but unsatisfied by B at time t. Let $Z_i^B(t) = \sum_{j\in Q_B(t)} p_{i,j}^B(t)$. When the algorithm B is the optimal solution (OPT) we set B to be O and if the algorithm is WSETF we set B to be A. For example $Q_A(t)$ is the set of released and unsatisfied jobs for WSETF at time t. Finally, for a set of possibly overlapping time intervals I, let $|I|$ denote the total length of their union.

3 Analysis Tools

In this section we introduce some useful tools that we use for our analysis. First we present our novel lower bound on the optimal solution. This lower bound is the key to our analysis and the main technical contribution of the paper. The left-hand-side of the inequality in the lemma has an arbitrary function $x(t) : \mathbb{R}^+ \to \mathbb{R}^+ \setminus \{0\}$, while the right-hand-side is simply a fractional cost of the schedule in consideration. This lower bound is inspired by one presented in [11]. However, the lower bound given in [11] involves substantially different terms, and is only for the ℓ_k-norms of flow time. Our proof is considerably different from [11], and perhaps simpler. Since this lower bound applies to any objective that fits into the general cost function framework, we believe that this lower bound will prove to be useful for a variety of scheduling problems. The assumption in the lemma that g is convex is crucial; the lemma is not true otherwise. The usefulness of this lemma will become apparent in the following two sections. We prove this lemma in Section 6 after we show the power of the lemma.

Lemma 1. *Let σ be a set of jobs on a single machine with speed s'. Let B be any feasible schedule and $B(\sigma)$ be the total weighted fractional cost of B with objective function g that is differentiable and convex (g' is non-decreasing), with $g(0) = 0$. Let $x(t) : \mathbb{R}^+ \to \mathbb{R}^+ \setminus \{0\}$ be any function of t. Let $p^B_{x,i}(t) = (\min(w_i x(t), p^B_i(t)) - q^B_i(t))^+$. Finally, let $Z^B_x(t) = \sum_{i \in Q_B(t)} p^B_{x,i}(t)$. Then,*

$$\int_{t=0}^{\infty} \frac{1}{x(t)} g(Z^B_x(t)/s')\mathrm{dt} \leq \frac{1}{s'} B(\sigma).$$

Next we show a property of WSETF that will be useful in relating the volume of work of unsatisfied jobs in WSETF's schedule to that of the optimal solution's schedule. By using this lemma we can bound the volume of jobs in the optimal solution's schedule and then appeal to the lower bound shown in the previous lemma. This lemma is somewhat similar to one shown for the algorithm Shortest-Remaining-Processing-Time (SRPT) [2,22]. However, we are able to get a stronger version of this lemma for WSETF.

Lemma 2. *Consider running WSETF using s-speed for some $s \geq 2$ on m identical machines and the optimal schedule at unit speed on m identical machines. For any job $i \in Q_A(t)$ and time t, it is the case that $Z^A_i(t) - Z^O_i(t) \leq 0$.*

Proof. For the sake of contradiction, let t be the earliest time such that $Z^A_i(t) - Z^O_i(t) > 0$. Let j be a job where $p^A_{i,j}(t) > p^O_{i,j}(t)$. Consider the interval $I = [r_j, t]$. Let I_j be the set of intervals where WSETF works on job j during I and let I'_j be the rest of the interval I. Knowing that $p^A_{i,j}(t) > p^O_{i,j}(t)$, we have that $|I_j| < \frac{1}{s}|I|$. If this fact were not true, then $q^A_j(t) = s|I_j| \geq |I|$, but since OPT has 1 speed, $q^O_j(t) \leq |I|$, and therefore $q^A_j(t) \geq q^O_j(t)$, a contradiction of the definition of job j. Hence, we know that $|I'_j| \geq (1 - \frac{1}{s})|I|$. At each time during I either WSETF is scheduling job j or all m machines in WSETF's schedule are busy scheduling jobs which contribute to $Z^A_i(t)$. Thus the total amount of

work done by WSETF during $|I|$ on jobs that contribute to $Z_i^A(t)$ is at least $q_j^A(t) + ms|I'_j| \geq ms(1 - \frac{1}{s})|I| = m(s-1)|I|$. The total amount of work OPT can do on jobs that contribute to $Z_i^O(t)$ is $m|I|$. Let S denote the set of jobs that arrive during I. The facts above imply that

$$\begin{aligned} Z_i^A(t) - Z_i^O(t) &\leq (Z_i^A(r_j) + \sum_{k \in S} p_{i,k} - m(s-1)|I|) - (Z_i^O(r_j) + \sum_{k \in S} p_{i,k} - m|I|) \\ &= (Z_i^A(r_j) - m(s-1)|I|) - (Z_i^O(r_j) - m|I|) \\ &\leq Z_i^A(r_j) - Z_i^O(r_j) \qquad [s \geq 2] \\ &\leq 0 \quad [t \text{ is the first time } Z_i^A(t) - Z_i^O(t) > 0 \text{ and } r_j < t]. \end{aligned}$$

□

4 Single Machine

We now show WSETF is 2-speed $O(1)$-competitive on a single processor for the fractional objective. We then derive Theorem 1. In Section 5, we extend our analysis to bound the performance of WSETF on identical machines as well when migration is allowed.

Assume that WSETF is given a speed $s \geq 2$. Notice that $Z_i^A(t)$ always decreases at a rate of s for all jobs $i \in Q_A(t)$ when $t \in [r_i, C_i]$. This is because $Z_i^A(t)$ is exactly the amount of remaining processing WSETF will do before job i is completed amongst jobs that have arrived by time t. Further, knowing that OPT has 1 speed, we see $Z_i^O(t)$ decreases at a rate of at most 1 at any time t. We know that by Lemma 2 $Z_i^A(r_i) - Z_i^O(r_i) \leq 0$. Using these facts, we derive for any time $t \in [r_i, C_i^A]$,

$$Z_i^A(t) - Z_i^O(t) \leq -(s-1) \cdot (t - r_i).$$

Therefore, $\frac{Z_i^O(t)}{s-1} \geq (t - r_i)$ for any $t \in [r_i, C_i^A]$. Let $a(t)$ denote the job that WSETF works on at time t. By the second definition, WSETF's fractional cost is

$$\int_{t=0}^{\infty} s \cdot \frac{w_{a(t)}}{p_{a(t)}} g(t - r_{a(t)}) \mathrm{dt} \leq s \int_{t=0}^{\infty} \frac{w_{a(t)}}{p_{a(t)}} g\Big(\frac{Z_{a(t)}^O(t)}{s-1}\Big) \mathrm{dt} \leq \frac{s}{s-1} \int_{t=0}^{\infty} \frac{w_{a(t)}}{p_{a(t)}} g(Z_{a(t)}^O(t)) \mathrm{dt}$$

The last inequality follows since $g(\cdot)$ is convex, $g(0) = 0$, and $\frac{1}{s-1} \leq 1$. By applying Lemma 1 with $x(t) = p_{a(t)}/w_{a(t)}$, $s' = 1$ and B being OPT's schedule, we have the following theorem.

Theorem 8. WSETF *is* s*-speed* $(1 + \frac{1}{s-1})$*-competitive for the fractional general cost function when* $s \geq 2$.

This theorem combined with Theorem 6 proves Theorem 1.

5 Multiple Identical Machines

Here we present the proof of Theorem 2. In the analysis of WSETF on a single machine, we bounded the cost of WSETF's schedule for the fractional objective to the cost of the optimal solution for the fractional objective. In the multiple machines case, we will not compare WSETF to the optimal solution for the fractional objective but rather compare to the cost of the optimal solution for the integral objective. We then invoke Theorem 7 to derive Theorem 2. We first consider an obvious lower bound on the optimal solution for the integral objective. For each job i, the best the optimal solution can do is to process job i immediately upon its arrival using one of its m unit speed machines. We know that the total integral cost of the optimal solution is at least

$$\sum_{i \in [n]} w_i g(p_i). \tag{3}$$

Similar to the single machine analysis, when a job is processed we charge the cost to the optimal solution. However, if a job i is processed at time t where $t - r_i \leq p_i$ we charge to the integral lower bound on the optimal solution above. If $t - r_i > p_i$, then we will invoke the lower bound on the optimal solution shown in Lemma 1 and use the fact that the an algorithm's fractional objective is always smaller than its integral objective.

Assume that WSETF is given speed $s \geq 3$. If job $i \in Q_A(t)$ is not processed by WSETF at time t, then there must exist at least m jobs in $Q_A(t)$ processed instead by WSETF at this time. Hence, for all jobs $i \in Q_A(t)$, the quantity $p_i^A(t) + Z_i^A(t)/m$ decreases at a rate of s during $[r_i, C_i^A]$. In contrast, the quantity $Z_i^O(t)/m$ decreases at a rate of at most 1 since OPT has m unit speed machines. Further, by Lemma 2, we know that $Z_i^A(r_i) - Z_i^O(r_i) \leq 0$, and $p_i^A(r_i) + Z_i^A(r_i) - Z_i^O(r_i) \leq p_i$. Using these facts we know for any job i and $t \in [r_i, C_i^A]$ that $p_i^A(t) + (Z_i^A(t) - Z_i^O(t))/m \leq p_i - (s-1)(t - r_i)$. Notice that if $t - r_i \geq p_i$, we have that $p_i^A(t) + (Z_i^A(t) - Z_i^O(t))/m \leq -(s-2)(t - r_i)$. Therefore, $t - r_i \leq \frac{Z_i^O(t)}{m(s-2)}$ when $t - r_i \geq p_i$.

Let $W(t)$ be the set of jobs that WSETF processes at time t. By definition, the value of WSETF's fractional objective is

$$s \int_{t=0}^{\infty} \sum_{i \in W(t)} \frac{w_i}{p_i} g(t - r_i) \mathrm{dt}.$$

We divide the set of jobs in $W(t)$ into two sets. The first is the set of 'young' jobs $W_y(t)$ which are the set of jobs $i \in W(t)$ where $t - r_i \leq p_i$. The other set is $W_o(t) = W(t) \setminus W_y(t)$ which is the set of 'old' jobs. Let OPT denote the optimal solution's integral cost. We see that WSETF's cost is at most the following.

$$
\begin{aligned}
s\int_{t=0}^{\infty}\sum_{i\in W(t)}\frac{w_i}{p_i}g(t-r_i)\mathrm{dt} &\le s\int_{t=0}^{\infty}\sum_{i\in W_y(t)}\frac{w_i}{p_i}g(t-r_i)\mathrm{dt} + s\int_{t=0}^{\infty}\sum_{i\in W_o(t)}\frac{w_i}{p_i}g(t-r_i)\mathrm{dt}\\
&\le \int_{t=0}^{\infty}\sum_{i\in W_y(t)} w_i\frac{s}{p_i}g(p_i)\mathrm{dt} + s\int_{t=0}^{\infty}\sum_{i\in W_o(t)}\frac{w_i}{p_i}g(t-r_i)\mathrm{dt}\\
&\le \sum_{i\in[n]} w_i g(p_i) + s\int_{t=0}^{\infty}\sum_{i\in W_o(t)}\frac{w_i}{p_i}g(t-r_i)\mathrm{dt}\\
&\le \mathrm{OPT} + s\int_{t=0}^{\infty}\sum_{i\in W_o(t)}\frac{w_i}{p_i}g\Big(\frac{Z_i^O(t)}{m(s-2)}\Big)\mathrm{dt}\\
&\quad \text{[by the lower bound of (3) on OPT]}\\
&\le \mathrm{OPT} + \frac{s}{s-2}\int_{t=0}^{\infty}\sum_{i\in W_o(t)}\frac{w_i}{p_i}g(Z_i^O(t)/m)\mathrm{dt}
\end{aligned}
$$

The third inequality holds since a job i can be in $W_y(t)$ only if i is processed by WSETF at time t, and job i can be processed by at most p_i before it is completed. More precisely, if i is in $W_y(t)$, then it is processed by $s\cdot\mathrm{dt}$ during time $[t, t+\mathrm{dt})$. Hence, $\int_{t=0}^{\infty}\mathbf{1}[i\in W_y(t)]\cdot s\cdot\mathrm{dt}\le p_i$, where $\mathbf{1}[i\in W_y(t)]$ denotes the 0-1 indicator variable such that $\mathbf{1}[i\in W_y(t)]=1$ if and only if $i\in W_y(t)$. The last inequality follows since $g(\cdot)$ is convex, $g(0)=0$, and $\frac{1}{s-2}\le 1$. We know that a single m-speed machine is always as powerful as m unit speed machines, because a m-speed machine can simulate m unit speed machines. Thus, we can assume OPT has a single m-speed machine. We apply Lemma 1 with $x(t)=p_i/w_i$ for each $i\in W_o(t)$, $s'=m$ and B being OPT's schedule. Knowing that $|W_o(t)|\le m$, we conclude that $\int_{t=0}^{\infty}\sum_{a\in W_o(t)}\frac{w_a}{p_a}g(Z_a^O(t)/m)$ is at most the optimal solution's fractional cost. Knowing that any algorithm's fractional cost is at most its integral cost, we conclude that WSETF's fractional cost with s-speed is at most $(2+\frac{2}{s-2})$ times the integral cost of the optimal solution when $s\ge 3$. Using Theorem 7, we derive Theorem 2.

6 Proof of the Main Lemma

In this section we prove Lemma 1.

Proof of [Lemma 1]

The intuition behind the lemma is that each instance of $Z_x^B(t)$ is composed of several infinitesimal job 'slices'. By integrating over how long these slices have left to live, we get an upper bound on $Z_x^B(t)$. We then argue that the integration over each slice's time alive is actually the fractional cost of that slice according to the second definition of the fractional objective. Recall $\beta_i^B(p)$ denotes the latest time t at which $p_i^B(t)=p$. For any time t, let

$$\Lambda_i(t)=\frac{w_i}{p_i}\int_{p=0}^{p_i^B(t)} g'(\beta_i^B(p)-t)\mathrm{dp},$$

and let $\Lambda(t) = \sum_{i \in Q_B(t)} \Lambda_i(t)$.

The proof of the lemma proceeds as follows. We first show a lower bound on $\Lambda(t)$ in terms of $\frac{1}{x(t)} g(Z_x^B(t)/s')$. Then we show an upper bound on $\Lambda(t)$ in terms of the fractional cost of B's schedule. This strategy allows us to relate $\frac{1}{x(t)} g(Z_x^B(t)/s')$ and B's cost. For the first part of the strategy, we prove that $\frac{s'}{x(t)} g(Z_x^B(t)/s') \leq \Lambda(t)$ at all times t. Consider any job $i \in Q_B(t)$ with $p_{x,i}^B(t) > 0$. Suppose $p_i \leq w_i x(t)$. Then,

$$\Lambda_i(t) = \frac{w_i}{p_i} \int_{p=0}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp} \geq \frac{1}{x(t)} \int_{p=p_i^B(t)-p_{x,i}^B(t)}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp}.$$

If $p_i > w_i x(t)$, then by definition of $p_{x,i}^B(t)$,

$$\begin{aligned}
\frac{p_i^B(t)}{p_{x,i}^B(t)} &\geq \frac{p_i^B(t) + q_i^B(t)}{p_{x,i}^B(t) + q_i^B(t)} \qquad [\text{Since } p_i^B(t) \geq p_{x,i}^B(t)] \\
&= \frac{p_i}{\min(w_i x(t), p_i^B(t)) - q_i^B(t))^+ + q_i^B(t)} \\
&\geq \frac{p_i}{(w_i x(t) - q_i^B(t)) + q_i^B(t)} \qquad [\text{Since } p_{x,i}^B(t) > 0] \\
&= \frac{p_i}{w_i x(t)}.
\end{aligned}$$

In this case,

$$\begin{aligned}
\Lambda_i(t) &= \frac{w_i}{p_i} \int_{p=0}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp} \\
&\geq \frac{p_i^B(t) w_i}{p_{x,i}^B(t) p_i} \int_{p=p_i^B(t)-p_{x,i}^B(t)}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp} \qquad [\text{Since } g \text{ is non-decreasing, convex}] \\
&\geq \frac{1}{x(t)} \int_{p=p_i^B(t)-p_{x,i}^B(t)}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dp} \qquad [\text{Since } p_i^B(t)/p_{x,i}^B(t) > p_i/(w_i x(t))].
\end{aligned} \tag{4}$$

In either case, $\Lambda_i(t)$ has a lower bound of quantity (4). By convexity of g, the lower bounds on $\Lambda_i(t)$ are minimized if B completes $p_{x,i}^B(t)$ units of i as quickly as possible for each job i. Schedule B runs at speed s', so we have

$$\Lambda(t) \geq \frac{1}{x(t)} \int_{p=0}^{Z_x^B(t)} g'(p/s')\mathrm{dp} = \frac{s'}{x(t)} \int_{p=0}^{Z_x^B(t)/s'} g'(p)\mathrm{dp} \geq \frac{s'}{x(t)} g(Z_x^B(t)/s').$$

This proves that lower bound on $\Lambda(t)$. Now we show an upper bound on $\Lambda(t)$ in terms of the B's fractional cost. We show $\int_{t=0}^{\infty} \Lambda(t)\mathrm{dt} \leq B(I)$. Fix a job i. We have

$$\int_{t=0}^{\infty} \Lambda_i(t)\mathrm{dt} = \int_{t=0}^{\infty} \frac{w_i}{p_i} \int_{p=0}^{p_i^B(t)} g'(\beta_i^B(p) - t)\mathrm{dpdt} = \frac{w_i}{p_i} \int_{p=0}^{p_i} \int_{t=0}^{\beta_i^B(p)} g'(t)\mathrm{dtdp}$$
$$= \frac{w_i}{p_i} \int_{p=0}^{p_i} g(\beta_i^B(p))\mathrm{dp}.$$

By summing over all jobs and using the definition of fractional flow time, we have that $\int_{t=0}^{\infty} \Lambda(t)\mathrm{dt} \leq B(I)$. Further, the given lower bound and upper bounds on $\int_{t=0}^{\infty} \Lambda(t)\mathrm{dt}$ show us that $\int_{t=0}^{\infty} \frac{s'}{x(t)} g(Z_x^B(t)/s')\mathrm{dt} \leq \int_{t=0}^{\infty} \Lambda(t)\mathrm{dt} \leq B(I)$, which proves the lemma. □

7 Lower Bounds

We now present the proof of Theorem 3. This lower bound extends a lower bound given in [1]. In [1], it was shown that no oblivious algorithm can be $O(1)$-competitive with speed less than $2-\epsilon$ for the general cost function. However, they assume that the cost function was possibly discontinuous and not convex. We show that their lower bound can be extended to the case where g is convex and continuous. This shows that WSETF is essentially the best oblivious algorithm one can hope for. In all the proofs that follow, we will consider a general cost function g that is continuous, non-decreasing, and convex. The function is also differentiable except at a single point. The function can be easily adapted so that it is differentiable over all points in $\mathbb{R}^+$.

Proof of [Theorem 3]: We appeal to Yao's Min-max Principle [26]. Let A be any deterministic online algorithm. Consider the cost function g and large constant c such that $g(F) = 2c(F - D)$ for $F > D$ and $g(F) = 0$ for $0 \leq F \leq D$. It is easy to see that g is continuous, non-decreasing, and convex. The constant D is hidden to A, and is set to 1 with probability $\frac{1}{2c(n+1)}$ and to $n+1$ with probability $1 - \frac{1}{2c(n+1)}$. Let $\mathcal{E}$ denote the event that $D = 1$. At time 0, one big job J_b of size $n+1$ is released. At each integer time $1 \leq t \leq n$, one unit sized job J_t is released. Here n is assumed to be sufficiently large. That is $n > \frac{12c}{\epsilon^2}$. Note that the event $\mathcal{E}$ has no effect on A's scheduling decision, since A is ignorant of the cost function.

Suppose the online algorithm A finishes the big job J_b by time $n+2$. Further, say the event $\mathcal{E}$ occurs; that is $D = 1$. Since $2n+1$ volume of jobs in total are released and A can process at most $(2-\epsilon)(n+2)$ amount of work during $[0, n+2]$, A has at least $2n+1-(2-\epsilon)(n+2) = \epsilon(n+2)-3$ volume of unit sized jobs unfinished at time $n+2$. A has total cost at least $2c(\epsilon(n+2)-3)^2/2 > c(\epsilon n)^2/2$. The inequality follows since $n > \frac{12c}{\epsilon^2}$. Knowing that $\Pr[\mathcal{E}] = \frac{1}{2c(n+1)}$, A has an expected cost greater than $\Omega(n)$. Now suppose A did not finish J_b by time $n+2$. Conditioned on $\neg\mathcal{E}$, A has cost at least $2c$. Hence A's expected cost is at least $2c(1 - \frac{1}{2c(n+1)}) > c$.

We now consider the adversary's schedule. Conditioned on $\mathcal{E}$ ($D = 1$), the adversary completes each unit sized job within one unit time and hence has a non-zero

cost only for J_b. The total cost is $2c(n+1)$. Conditioned on $\neg\mathcal{E}$ $(D = n+1)$, the adversary schedules jobs in a first in first out fashion thereby having cost 0. Hence the adversary's expected cost is $\frac{1}{2c(n+1)}(2c)(n+1) = 1$. Knowing that n is sufficiently larger than c, the claim follows since A has cost greater than c in expectation. □

Next we show a lower bound for any non-clairvoyant algorithm that knows g. In [1] it was shown that no algorithm can be $O(1)$-competitive for a general cost function with speed less than $7/6$. However, the cost function g used in the lower bound was neither continuous nor convex. We show that no algorithm can have a bounded competitive ratio if it is given a speed less than $\sqrt{2} > 7/6$ even if the function is continuous and convex but the algorithm is required to be non-clairvoyant.

Proof of [Theorem 4]: Let A be any non-clairvoyant deterministic online algorithm with speed s. Let the cost function g be defined as $g(F) = F - 10$ for $F > 10$ and $g(F) = 0$ otherwise. It is easy to verify that g is continuous, non-decreasing, and convex. At time $t = 0$, job J_1 of processing length 10 units and weight w_1 is released. At time $t = 10(\sqrt{2}-1)$, job J_2 of weight w_2 is released. Weights of these jobs will be set later. The processing time of job J_2 is set based on the algorithm's decisions, which can be done since the algorithm A is non-clairovyant.

Consider the amount of work done by A on the job J_2 by the time $t = 10$. Suppose algorithm A worked on J_2 for less than $10(\sqrt{2}-1)$ units by time $t = 10$. In this case, the adversary sets J_2's processing time to 10 units. The flow time of job J_2 in A's schedule is $(10-10(\sqrt{2}-1))+(10-10(\sqrt{2}-1))/s \geq 10+10(\sqrt{2}-1)\epsilon/(\sqrt{2}-\epsilon)$ when $s = \sqrt{2}-\epsilon$. Let $\epsilon' = 10(\sqrt{2}-1)\epsilon/(\sqrt{2}-\epsilon)$. Hence, A incurs a weighted flow time of $\epsilon' w_2$ towards J_2. The optimal solution works on J_2 the moment it arrives until its completion, so this job incurs no cost. The optimal solution processes J_1 partially before J_2 arrives and processes it until completion after job J_2 is completed. The largest flow time the optimal solution can have for J_1 is 20, so the optimal cost is upper bounded by $10w_1$. The competitive ratio of A $\frac{\epsilon' w_2}{10w_1}$ can be made arbitrarily large by setting w_2 to be much larger than w_1.

Now consider the case where A works on J_2 for $10(\sqrt{2}-1)$ units by time $t = 10$. In this case, the adversary sets the processing time of job J_2 to $10(\sqrt{2}-1)$. Therefore, A completes J_2 by time $t = 10$. However, A can not complete J_1 with flow time of at most 10 units, if given a speed of at most $\sqrt{2}-\epsilon$. Hence A incurs a cost of ϵw_1 towards flow time of J_1. It is easy to verify that for this input, the optimal solution first schedules J_1 until its completion and then processes job J_2 to completion. Hence, the optimal solution completes both the jobs with flow time of at most 10 units, incurring a cost of 0. Again, the competitive ratio is unbounded. □

Finally, we show a lower bound for any non-clairvoyant algorithm that knows g on m identical machines. We show that no algorithm can have a bounded competitive ratio when given speed less than $2-\frac{1}{m}$. Previously, the only previous

lower bounds for the general cost function on identical machines were lower bounds that carried over from the single machine setting.

Proof of [Theorem 5]: We use Yao's min-max principle. Let A be any non-clairvoyant deterministic online algorithm on m parallel machines with the speed $s = 2 - \epsilon$, for any $0 < \epsilon \leq 1$. Let $L > 1$ be a parameter and we take $m > \frac{1}{\epsilon}$. Let the cost function $g(F)$ be defined as follows: $g(F) = F - L$ for $F > L$ and $g(F) = 0$ otherwise. It is easy to verify that, g is continuous, non-decreasing, and convex. At time $t = 0$, $(m-1)L + 1$ jobs are released into the system, out of which $(m-1)L$ jobs have unit processing time and one job has processing time L. The adversary sets the job with processing time L uniformaly at random amongst all the jobs.

Consider the time $t = \frac{L(m-1)+1}{sm}$. At the time t, there exist a job j that has been processed to the extent of at most 1 unit by A since the most work A can do is $smt = L(m-1) + 1$, which is the total number of jobs. With probability $\frac{1}{L(m-1)+1}$, j has a processing time of L units. In the event that j has the processing time of L units, the earliest A can complete j is $t + \frac{L-1}{s} = \frac{L(m-1)+1}{sm} + \frac{L-1}{s} > L$ when L is sufficiently large and $s \leq 2 - \epsilon$ (note that $m > \frac{1}{\epsilon}$). In this case, j has a flow time greater than L time units. Therefore, in expectation A incurs a positive cost.

Let us now look at the adversary's schedule. Since the adversary knows the processing times of jobs, the adversary processes the job j of length L on a dedicated machine. The rest of the unit length jobs are processed on other machines. The adversary completes all the jobs by the time L and hence pays cost of 0. Therefore, the expected competitive ratio of the online algorithm A is unbounded. □

References

1. Im, S., Moseley, B., Pruhs, K.: Online scheduling with general cost functions. In: SODA, pp. 1254–1265 (2012)
2. Pruhs, K., Sgall, J., Torng, E.: Online Scheduling. In: Handbook of Scheduling: Algorithms, Models, and Performance Analysis (2004)
3. Bansal, N., Chan, H.L.: Weighted flow time does not admit o(1)-competitive algorithms. In: SODA, pp. 1238–1244 (2009)
4. Bender, M.A., Chakrabarti, S., Muthukrishnan, S.: Flow and stretch metrics for scheduling continuous job streams. In: SODA, pp. 270–279 (1998)
5. Chekuri, C., Im, S., Moseley, B.: Online scheduling to minimize maximum response time and maximum delay factor. Theory of Computing 8(1), 165–195 (2012)
6. Kalyanasundaram, B., Pruhs, K.: Speed is as powerful as clairvoyance. Journal of the ACM 47(4), 617–643 (2000)
7. Phillips, C.A., Stein, C., Torng, E., Wein, J.: Optimal time-critical scheduling via resource augmentation. Algorithmica 32(2), 163–200 (2002)
8. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Pruhs, K.: Online weighted flow time and deadline scheduling. Journal of Discrete Algorithms 4(3), 339–352 (2006)
9. Bansal, N., Pruhs, K.: Server scheduling to balance priorities, fairness, and average quality of service. SIAM J. Comput. 39(7), 3311–3335 (2010)

10. Edmonds, J., Im, S., Moseley, B.: Online scalable scheduling for the ℓ_k-norms of flow time without conservation of work. In: ACM-SIAM Symposium on Discrete Algorithms (2011)
11. Im, S., Moseley, B.: An online scalable algorithm for minimizing ℓ_k-norms of weighted flow time on unrelated machines. In: ACM-SIAM Symposium on Discrete Algorithms (2011)
12. Anand, S., Garg, N., Kumar, A.: Resource augmentation for weighted flow-time explained by dual fitting. In: SODA, pp. 1228–1241 (2012)
13. Azar, Y., Epstein, L., Richter, Y., Woeginger, G.J.: All-norm approximation algorithms. J. Algorithms 52(2), 120–133 (2004)
14. Kumar, V.S.A., Marathe, M.V., Parthasarathy, S., Srinivasan, A.: A unified approach to scheduling on unrelated parallel machines. J. ACM 56(5) (2009)
15. Leonardi, S., Raz, D.: Approximating total flow time on parallel machines. J. Comput. Syst. Sci. 73(6), 875–891 (2007)
16. Chekuri, C., Goel, A., Khanna, S., Kumar, A.: Multi-processor scheduling to minimize flow time with epsilon resource augmentation. In: STOC, pp. 363–372 (2004)
17. Awerbuch, B., Azar, Y., Leonardi, S., Regev, O.: Minimizing the flow time without migration. SIAM J. Comput. 31(5), 1370–1382 (2002)
18. Avrahami, N., Azar, Y.: Minimizing total flow time and total completion time with immediate dispatching. In: SPAA 2003: Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 11–18 (2003)
19. Becchetti, L., Leonardi, S.: Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. J. ACM 51(4), 517–539 (2004)
20. Chekuri, C., Khanna, S., Zhu, A.: Algorithms for minimizing weighted flow time. In: STOC, pp. 84–93 (2001)
21. Bussema, C., Torng, E.: Greedy multiprocessor server scheduling. Oper. Res. Lett. 34(4), 451–458 (2006)
22. Fox, K., Moseley, B.: Online scheduling on identical machines using srpt. In: SODA, pp. 120–128 (2011)
23. Edmonds, J., Pruhs, K.: Scalably scheduling processes with arbitrary speedup curves. In: ACM-SIAM Symposium on Discrete Algorithms, pp. 685–692 (2009)
24. Bansal, N., Krishnaswamy, R., Nagarajan, V.: Better scalable algorithms for broadcast scheduling. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part I. LNCS, vol. 6198, pp. 324–335. Springer, Heidelberg (2010)
25. Bansal, N., Pruhs, K.: The geometry of scheduling. In: IEEE Symposium on the Foundations of Computer Science, pp. 407–414 (2010)
26. Borodin, A., El-Yaniv, R.: On randomization in online computation. In: IEEE Conference on Computational Complexity, pp. 226–238 (1997)

Shrinking Maxima, Decreasing Costs: New Online Packing and Covering Problems

Pierre Fraigniaud[1], Magnús M. Halldórsson[2], Boaz Patt-Shamir[3], Dror Rawitz[3], and Adi Rosén[1]

[1] LIAFA, CNRS and University Paris Diderot, France
{pierre.fraigniaud,adiro}@liafa.univ-paris-diderot.fr
[2] School of Computer Science, Reykjavik University, 103 Reykjavik, Iceland
mmh@ru.is
[3] School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel
{boaz,rawitz}@eng.tau.ac.il

Abstract. We consider two new variants of online integer programs that are dual to each other. In the packing problem we are given a set of items and a collection of knapsack constraints over these items that are revealed over time in an online fashion. Upon arrival of a constraint we may need to remove several items (irrevocably) so as to maintain feasibility of the solution. Hence, the set of packed items becomes smaller over time. The goal is to maximize the number, or value, of packed items. The problem originates from a buffer-overflow model in communication networks, where items represent information units broken to multiple packets. The other problem considered is online covering: There is a universe we need to cover. Sets arrive online, and we must decide whether we take each set to the cover or give it up, so the number of sets in the solution grows over time. The cost of a solution is the total cost of sets taken, plus a penalty for each uncovered element. This problem is motivated by team formation, where the universe consists of skills, and sets represent candidates we may hire.

The packing problem was introduced in [8] for the special case where the matrix is binary; in this paper we extend the solution to general matrices with non-negative integer entries. The covering problem is introduced in this paper; we present matching upper and lower bounds on its competitive ratio.

1 Introduction

In this paper we study two related online problems based on the classic packing and covering integer programs. The first is a general packing problem called Online Packing Integer Programs (abbreviated opip). In this problem we are given a set of n items and a collection of knapsack constraints over these items. Initially the constraints are unknown and all items are considered packed. In each time step, a new constraint arrives, and the online algorithm needs to remove some items (irrevocably) so as to maintain feasibility of its solution. The goal is to maximize the number, or value, of packed items. Formally, the offline

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 158–172, 2013.

version of the problem we consider is expressed by the following linear integer program ($\mathbb{N}$ denotes the set of non-negative integers):

$$\begin{aligned} \max & \sum_{j=1}^{n} b_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \leq c_i \ \forall i \\ & x_j \leq p_j \qquad \forall j \\ & x_j \in \mathbb{N} \qquad \forall j \end{aligned} \tag{PIP}$$

We assume that $A \in \mathbb{N}^{m \times n}$ and $c \in \mathbb{N}^n$. The value of x_j represents the number of copies of item j that are packed, p_j is an upper bound on the number of copies of item j, b_j is the *benefit* obtained by packing item j, and c_i is the *capacity* of the ith constraint. The online character of OPIP is expressed by the following additional assumptions: (i) knapsack constraints arrive one by one, and (ii) the variables can only be decreased. The special case, where $A \in \{0,1\}^{m \times n}$ and $c = 1^n$ is known as Online Set Packing [8].

An LP-relaxation of (PIP) is obtained by replacing the integrality constraints by $x_j \geq 0$, for every j. It follows that the integral version of the dual of the LP-relaxation is:

$$\begin{aligned} \min & \sum_{i=1}^{m} c_i y_i + \sum_{j=1}^{n} p_j z_j \\ \text{s.t.} & \sum_{i=1}^{m} a_{ij} y_i + z_j \geq b_j \ \forall j \\ & y_i \in \mathbb{N} \qquad \forall i \\ & z_j \in \mathbb{N} \qquad \forall j \end{aligned} \tag{TF}$$

The program (TF) describes the second problem that is considered in this paper, called the Team Formation problem (for reasons that will become apparent below). In this problem we are given n elements, where element j has a covering requirement b_j and a penalty p_j. There are m sets, where the coverage of set i of element j is a_{ij} and its cost is c_i. The solution is a collection of the sets, where multiple copies of sets are allowed. The cost of a solution is the cost of selected sets plus the penalties for unsatisfied covering requirements. In (TF), the value of y_i represents the number of copies of i taken by the solution. Our online version of the Team Formation problem, denoted OTF, is as follows. Initially, the elements are uncovered—and hence incur a unit penalty per each unit of uncovered element. Sets with various coverage and cost arrive online. In each time step, a new set arrives, and the algorithm must decide how many copies of the arriving set to add to the solution. The goal is to minimize the total cost of sets taken plus penalties for uncovered elements.

Our main figure of merit, as is customary with online algorithms, is the *competitive ratio*: in the covering case, the ratio of cost incurred by the algorithm (expected cost if the algorithm is randomized) to the best possible cost for the

given instance, and in the packing case, the ratio between the benefit earned by the optimum solution to the (expected) benefit earned by the algorithm.

Motivation. The OTF problem is an abstraction of the following situation (corresponding to a binary matrix and binary requirements). We are embarking on a new project, which requires some n skills. The requirement for skill j can be satisfied by outsourcing for some cost p_j, or by hiring an employee who possesses skill j. The goal is to minimize the project cost under the following procedure: We interview candidates one by one. After each interview we know what are the skills of the candidate and what is the cost of hiring her, and then we must decide: do we hire the candidate? If we don't, we lose her forever.

The OPIP problem came out of the following natural networking situation [8]. High-level information units, called frames, may be too large to fit in a single network packet, in which case the frames are fragmented to multiple packets. As packets traverse the network, they may arrive at a bottleneck link which cannot deliver them all, giving rise to a basic online question: which packets to drop so as to maximize the number of frames that are completely delivered. If we ignore buffers, this question is precisely our version of OPIP: in each time step i, a burst of packets arrives, which corresponds to the ith constraint in (PIP): a_{ij} is the size of the packet from frame j that arrives at step i, and c_i is the total size that the link can deliver at time i.

Our problems appear unique in the literature of online computation in that solutions get progressively *smaller* with time. Traditionally, the initial solution is expected to be the empty set, and its value or cost only gets larger as the input is progressively presented. In our class of problems, some aspects of the input are known, inducing a naïve initial solution. The presented input progressively elucidates the structure of the instance, adding more constraints (in maximization problems) or providing increasing opportunities for cost reductions or optimizations (in minimization problems). In reality, often the issue is not what to include, but what to keep. We feel that this complementary viewpoint is natural and deserves further treatment.

Contribution and Results. The contributions of this paper are twofold. In the conceptual level, to the best of our knowledge we are the first to formalize the OTF problem (the OPIP problem was introduced in [8]).

On the technical level, we present new results for both the OPIP and the OTF problems. For OPIP, we extend the results of [8] from a binary matrix to the case of general non-negative integer demands. This is a useful extension when we consider our motivating network bottleneck scenario: it allows the algorithm to deal with packets of different size, while previous solutions were restricted to uniform-size packets. For the case of unit caps (i.e., $p = 1$), the competitive ratio of our algorithm is $O(C_{\max}\sqrt{\rho_{\max}})$, where $C_{\max}$ the maximal sum of entries in a column, and $\rho_{\max}$ is the maximal ratio of the sum of entries in a row i to its cap c_i. An additional $\sqrt{\max_j p_j}$ factor is incurred for non-unit cap. We remark that the extension is non-trivial, although it uses known techniques.

Regarding OTF, we prove matching upper and lower bounds on the competitive ratio: We show that even randomized algorithms cannot have competitive ratio better than $\Omega(\sqrt{\gamma})$, where γ is the maximal ratio, over all elements, between the highest and lowest cost of covering a given element. This result holds even for the case where the algorithm may discard a set from its running solution (but never takes back a set that was dismissed). On the other hand, we give a simple deterministic algorithm with a competitive ratio of $O(\sqrt{\gamma})$.

Related Work. Online packing was studied in the past, but traditionally the elements of the universe (equivalently, the constraints) were given ahead of time and sets arrive on-line (e.g., in [2]). In the similar vein, online set cover was defined in [1] as follows. A collection of sets is given ahead of time. Elements arrive online, and the algorithm is required to maintain a cover of the elements that arrived: if the arriving element is not already covered, then some set from the given collection must be added to the solution. Our problems have the complementary view of what's known in advance and what arrives online (see also [5]).

As mentioned above, our notion of OPIP was essentially introduced in [8]. Let us first review some results for the off-line packing problem PIP. The single constraint case ($m = 1$) is simply the KNAPSACK problem which is NP-hard and has an FPTAS [20,16]. If the number of constraints is constant, the offline version of PIP becomes the MULTI-DIMENSIONAL KNAPSACK problem that has a PTAS [11], while obtaining an FPTAS for it is NP-hard [17]. Raghavan and Thompson [19] used randomized rounding to obtain solutions whose benefit is $t_1 = \Omega(\text{OPT}/m^{1/\alpha})$ for PIP, where $\alpha = \min_j \min_i \frac{c_j}{a_{ij}}$. A solution of benefit $t_2 = \Omega(\text{OPT}/m^{1/(\alpha+1)})$ is also given for the case where $A \in \{0,1\}^{m\times n}$. (In this case $\alpha = \min_j c_j$.) Srinivasan [21] improved these results by obtaining solutions whose benefits are $\Omega(t_1^{\alpha/(\alpha-1)})$ and $\Omega(t_2^{\alpha/(\alpha-1)})$. Chekuri and Khanna [6] showed that, for every fixed integer α and fixed $\varepsilon > 0$, PIP with $c = \alpha^m$ and $A \in \{0,1\}^{m\times n}$ cannot be approximated within a factor of $m^{1/(\alpha+1)-\varepsilon}$, unless NP=ZPP. They also showed that PIP with uniform capacities cannot be approximated within a factor of $m^{1/(\alpha+1)-\varepsilon}$, unless NP=ZPP, even with a resource augmentation factor α. (In this case the solution x satisfies $Ax \leq \alpha c$.)

As mentioned before, the special case of PIP, where $A \in \{0,1\}^{m\times n}$ and $c = 1^n$ is known as SET PACKING. This problem is as hard as MAXIMUM INDEPENDENT SET even when all elements have degree 2 (i.e., A contains at most two non-zero entries in each row), and therefore cannot be approximated to within a factor of $O(n^{1-\epsilon})$, for any $\varepsilon > 0$ [14]. In terms of the number of elements (constraints, in PIP terms) SET PACKING packing is $O(\sqrt{m})$-approximable, and hard to approximate within $m^{1/2-\varepsilon}$, for any $\varepsilon > 0$ [13]. When set sizes are at most k (A contains at most k non-zero entries in each column), it is approximable to within $(k+1)/3 + \varepsilon$, for any $\varepsilon > 0$ [7], and within $(k+1)/2$ in the weighted case [4], but known to be hard to approximate to within $o(k/\log k)$-factor [15].

OPIP was introduced in [8], assuming that the matrix is binary, namely each set requires either one or zero copies of each element. In [8], a randomized algorithm was given for that case, obtaining competitive ratio of $O(k\sqrt{\sigma})$, where k is the

maximal set size and σ is the maximal ratio, over all elements, between the number of sets containing that element to the number of its copies. In OPIP terms this bound is $O(C_{\max}\sqrt{\rho_{\max}})$. A nearly matching lower bound of $\tilde{\Omega}(k\sqrt{\sigma})$ was also given. Subsequent work extended these results to allow for redundancy [18], i.e., when the benefit of a set is earned by the algorithm even if up to a $\beta < 1$ fraction of its elements are not assigned to it.

Previously, the online packing problem where sets arrive online and constraints are fixed was defined in [2]. They give an algorithm with competitive ratio $O(\log n)$ assuming that no set requires more than a $1/\log n$ fraction of the cap of any element. A matching lower bound shows that this requirement is necessary to obtain a polylogarithmic competitive ratio.

Regarding team formation, we are unaware of any prior formalization of the problem, let alone analysis. The online cover problem defined in [1] has an algorithm with competitive ratio $O(\log n \log m)$. Another related problem is the secretary problem (see, e.g., [12,10]; some more recent results and references can be found in [9,3]). In this family of problems a n candidates arrive in random order (or with random value), and the goal is to pick k of them (classically, $k = 1$) which optimizes some function of the value set, such as the probability of picking the candidates with the top k values, or the average rank of selected candidates. The difficulty, similarly to our OTF formulation, is that the decision regarding each candidate must be taken immediately upon her arrival. However, the stipulation that the input is random makes the secretary problem very different from OTF. Another difference is that unlike OTF, the number of candidates to pick is set in advance.

Paper Organization. The remainder of this paper is organized as follows. In Section 2 we introduce some notation. In Section 3 we describe our online algorithm for OPIP, and in Section 4 we consider OTF.

2 Preliminaries

In this section we define our notation. Given a matrix $A \in \mathbb{N}^{m\times n}$, let $R(i) = \sum_j a_{ij}$ be the sum of entries in the ith row, and let $C(j) = \sum_i a_{ij}$ be the sum of entries in the jth column. Denote $R_{\max} = \max_i R(i)$ and $C_{\max} = \max_j C(j)$.

Given an OPIP instance, define $\rho(i) = R(i)/c_i$. Observe that if $\sum_j a_{ij} \le c_i$ for some i, then constraint i is redundant. Hence we assume w.l.o.g. that $\sum_j a_{ij} > c_i$ for every i, which means that $\rho(i) > 1$, for every i. We assume hereafter that $\mathrm{lcm}(a_{i1}, \ldots, a_{in}, c_i) = 1$, for every i. This does not change $\rho(i)$, but it may decrease $C_{\max}$ and our bound on the competitive ratio. On the other extreme, we assume that $a_{ij} \le c_i$ for every i and j: if $a_{ij} > c_i$ then item j is not a member in any feasible solution.

Given a subset of items J and a constraint i, let $J(i) = \{j \in J : a_{ij} > 0\}$ be the subset of items from J that participate in constraint i. For example, if OPT is the set of items in some fixed optimal solution, then $\mathrm{OPT}(i)$ denotes the items in OPT that are active in constraint i. Also, let $R_J(i) = \sum_{j\in J} a_{ij}$.

Given a subset of items J, let $b(J) = \sum_{j \in J} b_j$. Also, we define the *normalized benefit* of a constraint i as $\bar{b}(i) = \sum_j a_{ij} \cdot b_j$.

Recall that $\rho(i) = \frac{R(i)}{c_i} = \frac{1}{c_i} \sum_j a_{ij}$. Given an OTF instance, $\sum_j a_{ij}$ is the coverage potential of a single copy of set i, while c_i is the cost of set i. Hence, $1/\rho(i)$ stands for the cost per unit of coverage that may be covered by i. We denote

$$\gamma_j = \max \left\{ p_j \cdot \max_{i:a_{ij}>0} \rho(i), 1 \right\} .$$

In other words, γ_j is the ratio between the penalty for not covering a unit of coverage of j and the minimum possible cost per unit of coverage that may be obtained to cover j. Also, denote $\gamma_{\max} = \max_j \gamma_j$.

3 Online Packing Integer Programs

In this section we describe a randomized algorithm for OPIP with unit caps, namely for the case where $p_j = 1$, for every j. The competitive ratio of our algorithm is $2C_{\max}\sqrt{\rho_{\max}}$. We note that the algorithm is a slight generalization of the algorithm given in [8], which allows us to deal with non-binary instances. We note that one may solve the general case by treating each item j as p_j items, but this simplistic approach results in an additional multiplicative factor of $\sqrt{\max_j p_j}$ to the competitive ratio.

Random Variables. For $w > 0$, let $D_w : \mathbb{R} \to [0,1]$ be a (cumulative) distribution function of a random variable Z that is defined by

$$D_w(z) \;=\; \Pr[Z \le z] \;=\; \begin{cases} 0 & \text{if } z < 0; \\ z^w & \text{if } 0 \le z < 1; \\ 1 & \text{if } 1 \le z. \end{cases}$$

Note that D_1 is the uniform distribution over $[0,1]$ and, in general, for a positive integer q, D_q is the distribution of the maximum of q independent and identically distributed variables, each uniformly distributed over $[0,1]$.

Algorithm RP. For each item j, we independently choose a random priority $p(j) \in [0,1]$ with distribution D_{b_j}. When constraint i arrives, we construct c_i subsets $S_{i1}, \ldots, S_{ic_i}$ as follows. Each item j chooses a_{ij} subsets at random. Then, for each subset $S_{i\ell}$, $\ell \in \{1, \ldots, c_i\}$, we reject all items but the one with the highest priority. Observe that an item survives only if it has the highest priority in all of its chosen sets.

Intuitively, the approach is to prefer items with high priority. In the special case where $a_{ij} \in \{0,1\}$, one may simply choose the c_i items with highest priority. A somewhat more subtle approach, based on a reduction to the unit capacity case is used in [8]: Items are randomly partitioned into c_i equal-size subsets; from each subset only the top priority item survives. Our Algorithm RP extends this approach: we construct c_i subsets whose expected sizes are equal, such that item j is contained in exactly a_{ij} subsets.

Analysis. Observe that each subset $S_{i\ell}$ induces the following constraint: $\sum_{j \in S_{i\ell}} x_j \leq 1$. Hence, we construct a new uniform capacity OPIP instance by defining the matrix $A' \in \{0,1\}^{(\sum_i c_i) \times n}$ as follows: $a_{\sum_{t<i} c_k + \ell, j} = 1$ if and only if $j \in S_{i\ell}$. Each row of A' corresponds to one of the random constraints created by the algorithm.

Observation 1. *$C(j) = C'(j)$, for every j, and $\mathbb{E}[R'(\sum_{t<i} c_t + \ell)] = \rho(i)$, for every i and ℓ.*

Proof. $C(j) = C'(j)$, since the item j appears in a_{ij} new constraints with coefficient 1, for every such constraint i. Each item j participates in the ℓth new constraint corresponding to original constraint i with probability a_{ij}/c_i. Hence,

$$\mathbb{E}[R'(\sum_{t<i} c_t + \ell)] = \sum_j \mathbb{E}[a'_{\sum_{t<i} c_t + \ell, j}] = \sum_i \frac{a_{ij}}{c_i} = \frac{R(i)}{c_i} \ .$$

□

Let $N[j]$ denote the items that are in conflict with j, namely

$$N[j] = \{k : \exists i, \ell \text{ s.t. } j, k \in S_{i\ell}\} \ .$$

Notice that $j \in N[j]$. We also define $N(j) = N[j] \setminus \{j\}$. Clearly, item j is satisfied by the algorithm if and only if its priority is higher than that of all other items with whom it competes, i.e., if $p(j) > p(k)$, for every $k \in N(j)$.

First, we consider the probability of satisfying an item j.

Lemma 2. $\Pr[p(j) > \max\{p(k) : k \in N(j)\}] = \mathbb{E}\left[\frac{b_j}{b(N[j])}\right]$.

Proof. Supposed that $N(j) = N$ and let $p_{\max} = \max\{p(k) : k \in N\}$. Then, for $z \in [0,1]$ we have

$$\Pr[p_{\max} < z] \ = \ \prod_{k \in N} \Pr[p(k) < z] \ = \ \prod_{k \in N} z^{b_k} \ = \ z^{\sum_{k \in N} b_k} \ = \ z^{b(N)} \ ,$$

that is, $p_{\max}$ has distribution $D_{b(N)}$. Hence,

$$\Pr[p(j) > p_{\max}] \ = \ \int_0^1 \Pr[p_{\max} < z] \cdot f_{p(j)}(z) dz \ = \ \int_0^1 z^{b(N)} \cdot b_j z^{b_j - 1} dz$$
$$= \ \frac{b_j}{b(N) + b_j} \ .$$

It follows that

$$\begin{aligned} &\Pr[p(j) > \max\{p(k) : k \in N(j)\}] \\ &= \ \sum_N \Pr[N(j) = N] \cdot \Pr[p(j) > \max\{p(k) : k \in N\} | N(j) = N] \\ &= \ \sum_N \Pr[N(j) = N] \cdot \frac{b_j}{b(N) + b_j} \\ &= \ \mathbb{E}\left[\frac{b_j}{b(N(j)) + b_j}\right] \end{aligned}$$

as required. □

Next, we provide a lower bound on the expected performance of the algorithm.

Lemma 3. *For any subset of items J, $\mathbb{E}[b(\mathrm{RP})] \geq \frac{b(J)^2}{\mathbb{E}\left[\sum_{j\in J} b(N[j])\right]}$.*

Proof. By Lemma 2, $\Pr[j \in \mathrm{RP}] = \mathbb{E}\left[\frac{b_j}{b(N[j])}\right]$. Thus, by linearity of expectation, we obtain

$$\mathbb{E}[b(\mathrm{RP})] \;=\; \sum_{j\in J} b_j \cdot \mathbb{E}\left[\frac{b_j}{b(N[j])}\right] \;=\; \mathbb{E}\left[\sum_{j\in J} \frac{b_j^2}{b(N[j])}\right] \;\geq\; \mathbb{E}\left[\frac{\left(\sum_{j\in J} b_j\right)^2}{\sum_{j\in J} b(N[j])}\right] ,$$

where the inequality is due to the following consequence of the Cauchy-Schwarz inequality (with b_j for α_j and $b(N[j])$ for β_j): for positive reals $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$, we have $\sum_j \frac{\alpha_j^2}{\beta_j} \geq \frac{\left(\sum_j \alpha_j\right)^2}{\sum_j \beta_j}$. Jensen's inequality (for a non-negative random variable X, $\mathbb{E}\left[\frac{1}{X}\right] \geq \frac{1}{\mathbb{E}[X]}$) implies that

$$\mathbb{E}[b(\mathrm{RP})] \;\geq\; \mathbb{E}\left[\frac{\left(\sum_{j\in J} b_j\right)^2}{\sum_{j\in J} b(N[j])}\right] \;\geq\; \frac{\left(\sum_{j\in J} b_j\right)^2}{\mathbb{E}\left[\sum_{j\in J} b(N[j])\right]} ,$$

and the lemma follows. □

Our next step is to bound $\sum_{j\in J} b(N[j])$.

Lemma 4. *Let J be a subset of items. Then, $\sum_{j\in J} b(N[j]) \;\leq\; \sum_{i=1}^{m'} R'_J(i)\bar{b}'(i)$.*

Proof. Observe that

$$\begin{aligned}
\sum_{j\in J} b(N[j]) &= \sum_{j\in J} \sum_{k\in N[j]} b_k \\
&\leq \sum_{j\in J} \sum_{(i,\ell):j\in S_{i\ell}} \sum_{k\in S_{i\ell}} b_k \\
&= \sum_{j\in J} \sum_{(i,\ell):j\in S_{i\ell}} b(S_{i\ell}) \\
&= \sum_{i=1}^{m} \sum_{\ell=1}^{c_i} |S_{i\ell} \cap J| \cdot b(S_{i\ell}) = \sum_{i=1}^{m'} R'_J(i)\bar{b}'(i) ,
\end{aligned}$$

as required. □

To complete the analysis we find appropriate upper bounds for the denominator when $J = [n]$ and when $J = \mathrm{OPT}$.

Lemma 5.

$$\mathbb{E}\left[\sum_{i=1}^{m'} R'_{[n]}(i)\bar{b}'(i)\right] < 2\sum_{i=1}^{m} \rho(i)\bar{b}(i) , \tag{1}$$

$$\mathbb{E}\left[\sum_{i=1}^{m'} R'_{\mathrm{OPT}}(i)\bar{b}'(i)\right] \leq \sum_{j\in[n]} C(j)b_j + \sum_{j\in\mathrm{OPT}} C(j)b_j \leq 2\sum_{j\in[n]} C(j)b_j . \tag{2}$$

Proof. Consider $i' \in [m']$ that corresponds to the ℓth new constraint of original constraint i, and two items $j \neq k$. We have that

$$\Pr[j,k \in S_{i\ell}] = \Pr[j \in S_{i\ell}] \cdot \Pr[k \in S_{i\ell}] = \frac{a_{ij}}{c_i} \cdot \frac{a_{ik}}{c_i} \ ,$$

due to the independence of the random choices of j and k. Hence, for $i \in [m]$ we have that

$$\begin{aligned}
\mathbb{E}\left[\sum_{\ell=1}^{c_i} R'_J\left(\sum_{t<i} c_t + \ell\right)\bar{b}'\left(\sum_{t<i} c_t + \ell\right)\right] &= \sum_{j\in J(i)}\sum_{k}\sum_{\ell=1}^{c_i} b_k \Pr[j,k \in S_{i\ell}] \\
&= \sum_{j\in J(i)} \frac{a_{ij}}{c_i} \sum_{k\neq j} c_i b_k \frac{a_{ik}}{c_i} + \sum_{j\in J(i)} c_i b_j \frac{a_{ij}}{c_i} \\
&\leq \sum_{j\in J(i)} \frac{a_{ij}}{c_i} \cdot \bar{b}(i) + \sum_{j\in J(i)} a_{ij} \cdot b_j \\
&\leq \rho_J(i)\bar{b}(i) + \bar{b}_J(i) \ .
\end{aligned}$$

It follows that

$$\mathbb{E}\left[\sum_{i=1}^{m'} R'_J(i)\bar{b}'(i)\right] \leq \sum_i \rho_J(i)\bar{b}(i) + \sum_i \bar{b}_J(i) \ . \tag{3}$$

Since $\rho(i) > 1$, for every i, Inequality (1) is obtained by assigning $J = [n]$ in (3).

To prove Inequality (2) we assign $J = \text{OPT}$. In this case, $\rho_{\text{OPT}}(i) \leq 1$, for every i, since OPT is a feasible solution. Hence

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{m'} R'_{\text{OPT}}(i)\bar{b}'(i)\right] &\leq \sum_i \bar{b}(i) + \sum_i \bar{b}_{\text{OPT}}(i) \\
&= \sum_i \sum_j a_{ij} b_j + \sum_i \sum_{j\in\text{OPT}} a_{ij} b_j \\
&= \sum_j b_i \sum_i a_{ij} + \sum_{j\in\text{OPT}} b_j \sum_i a_{ij} \\
&= \sum_j b_j C(j) + \sum_{j\in\text{OPT}} b_j C(j) \ ,
\end{aligned}$$

and the lemma follows. □

Lemma 5 implies that

Theorem 1.

$$\mathbb{E}[b(\text{RP})] \geq \max\left\{\frac{b([n])^2}{2\sum_i \rho(i)\bar{b}(i)}, \frac{b(\text{OPT})^2}{2\sum_j C(j)b_j}\right\} \geq \frac{b([n])b(\text{OPT})}{2\sqrt{\sum_i \rho(i)\bar{b}(i) \cdot \sum_j C(j)b_j}} \ .$$

Theorem 1 implies the following:

Corollary 1. *There is an* OPIP *algorithm with competitive ratio at most* $2C_{\max}\sqrt{\rho_{\max}}$.

Proof.

$$\sum_i \rho(i)\bar{b}(i) = \rho_{\max}\sum_i\sum_j a_{ij}b_j = \rho_{\max}\sum_j b_j C(j) \leq \rho_{\max} b([n])C_{\max} \ ,$$

and $\sum_j C(j)b_j \leq C_{\max}b([n])$. Hence,

$$\mathbb{E}[b(\mathrm{RP})] \ \geq \ \frac{b([n])b(\mathrm{OPT})}{2\sqrt{\rho_{\max}b([n])C_{\max}\cdot C_{\max}b([n])}} \ = \ \frac{b(\mathrm{OPT})}{2C_{\max}\sqrt{\rho_{\max}}} \ ,$$

and we are done. □

4 Competitive Team Formation

In this section we provide a deterministic online algorithm for OTF and a matching lower bound that holds even for randomized algorithms. Furthermore, our lower bound holds for a more general case, where the commitment of the online algorithm is only "one way" in the following sense. Once a set is dismissed it cannot be recruited again, but a set in the solution at one point may be thrown out of the solution later.

4.1 An Online Algorithm

Our algorithm generates a monotonically growing collection of sets based on a simple deterministic threshold rule. Recall that γ is the maximal ratio, over all elements, between the highest and lowest cost of covering a given element. Algorithm THRESHOLD assumes knowledge of γ and works as follows. Let y be the set vector that is constructed by THRESHOLD. Also, define $z_j^i = \max\left\{b_j - \sum_{\ell<i} a_{\ell j}y_\ell, 0\right\}$, namely z_j^i is the amount of missing coverage for element j. Note that z_j^i is monotone non-increasing with i.

Upon arrival of a new candidate i, assign $y_i \leftarrow v$, such that v is the maximum integer that satisfies

$$v \cdot c_i \leq \frac{\sum_j \min\left\{v \cdot a_{ij}, z_j^{i-1}\right\} \cdot p_j}{\sqrt{\gamma}} \ . \tag{4}$$

Intuitively, we take the maximum possible number of units of set i that allows us to save a factor of at least $\sqrt{\gamma}$ over the penalties it replaces. Note that $\min\{va_{ij}, z_j^{i-1}\}$ is the amount of coverage v copies of set i add to element j. Hence, the total amount of penalties that are saved by v copies of set i is $\sum_j \min\{va_{ij}, z_j^{i-1}\}p_j$. Also notice that v is well-defined because (4) is always satisfied by $v = 0$.

We show that the competitive ratio of THRESHOLD is at most $2\sqrt{\gamma}$.

Theorem 2. *Let (y, z) be the solution computed by Algorithm* THRESHOLD*, and let (y^*, z^*) be an optimal (integral) solution. Then,*

$$\sum_i c_i y_i + \sum_j p_j z_j \leq 2\sqrt{\gamma} \sum_i c_i y_i^* + (1 + 1/\sqrt{\gamma}) \sum_j p_j z_j^* \ .$$

Proof. We first bound $\sum_i c_i y_i$. Using condition (4), we then have that

$$\begin{aligned}
\sum_i c_i y_i &\leq \frac{1}{\sqrt{\gamma}} \sum_i \sum_j \min\{a_{ij} y_i, z_j^{i-1}\} \cdot p_j \\
&= \frac{1}{\sqrt{\gamma}} \sum_j p_j \sum_i \min\{a_{ij} y_i, z_j^{i-1}\} \\
&\leq \frac{1}{\sqrt{\gamma}} \sum_j p_j b_j \ ,
\end{aligned}$$

where the second inequality follows since $\min\{a_{ij} y_i, z_j^{i-1}\}$ is the amount of coverage that is added to j in the ith round, and therefore the total coverage of j, $\sum_i \min\{a_{ij} y_i, z_j^{i-1}\}$, is at most $b_j - z_j \leq b_j$. On the other hand, by the definition of γ_j we have that $\gamma_j \geq p_j \cdot \rho(i) = p_j \frac{R(i)}{c_i}$, for any i such that $a_{ij} > 0$. Hence,

$$\begin{aligned}
\sum_i c_i y_i^* &= \sum_i \frac{c_i}{R(i)} y_i^* \sum_j a_{ij} \\
&\geq \sum_i \sum_j y_i^* \frac{p_j}{\gamma_j} a_{ij} \\
&\geq \frac{1}{\gamma} \sum_j p_j \sum_i y_i^* a_{ij} \\
&\geq \frac{1}{\gamma} \sum_j p_j (b_j - z_j^*) \ .
\end{aligned}$$

It follows that

$$\begin{aligned}
\sum_i c_i y_i &\leq \frac{1}{\sqrt{\gamma}} \sum_j p_j b_j \\
&= \frac{1}{\sqrt{\gamma}} \left(\sum_j p_j (b_j - z_j^*) + \sum_j p_j z_j^* \right) \\
&\leq \sqrt{\gamma} \sum_i c_i y_i^* + \frac{1}{\sqrt{\gamma}} \sum_j p_j z_j^*
\end{aligned}$$

Next, we turn to bound the penalties that (y, z) pays and (y^*, z^*) does not pay, namely we bound $\sum_j p_j \max\{z_j - z_j^*, 0\}$. Define $\Delta_i = \max\{y_i^* - y_i, 0\}$. If $\Delta = 0$, then $z_j \leq z_j^*$, for every j, and we are done. Otherwise, let i be an index such

that $\Delta_i > 0$. Due to condition (4) in the ith step, we have that

$$c_i y_i \leq \frac{\sum_j \min\{a_{ij} y_i, z_j^{i-1}\} \cdot p_j}{\sqrt{\gamma}}$$

while

$$c_i y_i^* > \frac{\sum_j \min\{a_{ij} y_i^*, z_j^{i-1}\} \cdot p_j}{\sqrt{\gamma}} \ .$$

Observe that j's coverage increases by $\min\{a_{ij} y_i, z_j^{i-1}\} = z_j^i - z_j^{i-1}$ in the ith step. If we further increase y_i to y_i^* we may gain $\min\{\Delta_i a_{ij}, z_j^i\}$ additional coverage for item j. Hence,

$$c_i \Delta_i \ = \ c_i y_i^* - c_i y_i \ > \ \frac{\sum_j \min\{a_{ij} \Delta_i, z_j^i\} \cdot p_j}{\sqrt{\gamma}} \ \geq \ \frac{\sum_j \min\{a_{ij} \Delta_i, z_j\} \cdot p_j}{\sqrt{\gamma}} \ .$$

It follows that

$$\begin{aligned} \sqrt{\gamma} \sum_i c_i \Delta_i &> \sum_i \sum_j \min\{a_{ij} \Delta_i, z_j\} \cdot p_j \\ &\geq \sum_j p_j \min\left\{ \sum_i a_{ij} \Delta_i, z_j \right\} \\ &\geq \sum_j p_j \max\{z_j - z_j^*, 0\} \ , \end{aligned}$$

where the last inequality follows from the fact that $y + \Delta \geq y^*$ and therefore Δ covers at least $\max\{z_j - z_j^*, 0\}$, for every j. Hence,

$$\sum_j p_j \max\{z_j - z_j^*, 0\} \ \leq \ \sqrt{\gamma} \sum_i c_i \Delta_i \ \leq \ \sqrt{\gamma} \sum_i c_i y_i^* \ .$$

Putting it all together, we get that

$$\begin{aligned} \sum_i c_i y_i + \sum_j p_j z_j \ &\leq \ \sum_i c_i y_i + \sum_j p_j z_j^* + \sum_j p_j \max\{z_j - z_j^*, 0\} \\ &\leq \ 2\sqrt{\gamma} \sum_i c_i y_i^* + (1 + 1/\sqrt{\gamma}) \sum_j p_j z_j^* \ , \end{aligned}$$

as required. □

This leads us to an upper bound on the competitive ratio.

Corollary 2. *Algorithm* Threshold *is* $2\sqrt{\gamma}$*-competitive.*

We note that the same approach would work for the variant of otf in which there is an upper bound u_i on the number of copies of set i that can be used, i.e., $y_i \leq u_i$. In this case the value of v in condition (4) is also bounded by u_i. The rest of the details are omitted.

4.2 A Lower Bound

In this section we present a matching lower bound, which holds for randomized algorithms, and even for the case where the algorithm may discard a set from its running solution (but never takes back a set that was dismissed).

We start with a couple of simple constructions. In the first the input consists of sets of size one, and in the second all costs and penalties are the same.

Theorem 3. *The competitive ratio of any randomized algorithm for* OTF *is* $\Omega(\sqrt{\gamma})$. *This bound holds for inputs with only two elements and sets of size one.*

Proof. Let ALG be a randomized algorithm. Consider an input sequence consisting of two or three elements with unit covering requirement and penalty p. The arrival sequence is composed of two or three sets. The first set to arrive is $\{1\}$ of cost 1. (The goal of the first set is to make sure that the ratio between the penalty and the minimum cost is p.) The second set is $\{2\}$ of cost $\sqrt{p}$. If ALG takes this set with probability less than half, then the sequence ends; otherwise, the third set $\{2\}$ of cost 1 arrives.

In the first case the optimal cost is $1 + \sqrt{p}$, while ALG pays at least $1 + \frac{1}{2}p$. Otherwise, the optimal cost is 2, while ALG pays at least $1 + \frac{1}{2}\sqrt{p}$. Notice that we may repeat the second part of this sequence as many times as needed. Finally, notice that $\gamma = p$. □

Theorem 4. *The competitive ratio of any randomized online algorithm is at least* $\Omega(\sqrt{\gamma})$. *This bound holds for inputs with unit costs and penalties.*

Proof. Let ALG be a randomized algorithm. Assume unit penalties and unit coverage requirements. Consider the input sequence that starts with $\sqrt{n}$ candidates, each with $\sqrt{n}$ fresh skills and cost 1. Let ℓ be the expected number of candidates ALG takes from this sequence. If $\ell < \sqrt{n}/2$, this is the whole input. In this case the expected cost of ALG is at least $\frac{1}{2}n$, whereas the optimal cost is $\sqrt{n}$. If $\ell \geq \frac{1}{2}\sqrt{n}$, then we add an omnipotent candidate (who has all skills) at the end, with cost 1. It follows that ALG pays at least $\frac{1}{2}\sqrt{n}$ in expectation, while OPT pays only 1. Finally, notice that $\gamma = n$. □

Next, we give a lower bound construction that applies to the more general setting in which the algorithm may discard a set from its solution.

Theorem 5. *The competitive ratio of any randomized algorithm for* OTF *is* $\Omega(\sqrt{\gamma})$. *This bound holds even if the algorithm is allowed to discard sets. Furthermore, it holds also in the binary case, where all demands, coverages, penalties and costs are either* 0 *or* 1.

Proof. Our lower bound construction uses affine planes defined as follows. Let $n = q^2$, where q is prime. In our construction, each pair $(a, b) \in \mathbb{Z}_q \times \mathbb{Z}_q$ corresponds to an element. Sets will correspond to lines: a line in this finite geometry is a collection of pairs $(x, y) \in \mathbb{Z}_q \times \mathbb{Z}_q$ satisfying either $y \equiv ax + b \pmod{q}$,

for some given $a, b \in \mathbb{Z}_q$, or of the form $(c, *)$ for some given $c \in \mathbb{Z}_q$. There are $q^2 + q = \Theta(n)$ such lines.

The important properties we use are the following:

1. All points can be covered by q disjoint (parallel) lines.
2. Two lines that intersect in more than a single point are necessarily identical.

We now describe the lower bound scenario. The elements correspond (in a 1-1 fashion) to the points in the affine plane. All elements have unit penalty and unit covering requirement, i.e., $p_j = 1$ and $b_j = 1$, for every j. The input sequence starts with a sequence of q^2+q sets corresponding to all distinct lines of the plane, each with unit cost. Fix any randomized online algorithm ALG. We proceed by cases, depending on the expected number r of these sets that ALG retains at this point. If $r \leq \sqrt{n}/2$ or $r > n/2$, then we are already done: at this time the cost to the algorithm is $\Omega(n)$ (due either to penalties or to the cost of sets retained), while the optimal cost at this time is $\sqrt{n}$ by virtue of Property (1) above.

Otherwise, $\sqrt{n}/2 < r \leq n/2$. Let L be a line chosen uniformly at random. The probability that L is retained by the algorithm is at most $1/2$, since $r \leq n/2$. We now extend the input sequence by one more set $L^c \stackrel{\text{def}}{=} \{1, ..., n\} \setminus L$, and assign L^c unit cost. Note that by Property (2), if L is not retained by the algorithm, then the number of other lines that cover the points of L cannot be smaller than $|L| = \sqrt{n}$, and hence the expected cost of ALG due only to the points of L (either by covering set costs or by incurred penalties) is at least $\sqrt{n}/2$. Obviously, throwing out any set from the solution at this time will not help to reduce the cost. On the other hand, the optimal solution to this scenario is the sets L and L^c, whose cost is 2, and hence the competitive ratio is at least $\Omega(\sqrt{n})$. □

Remarks. First, we note that in the proof above, the unit-cost set L^c can be replaced by $\sqrt{n} - 1$ sets, where each set covers $\sqrt{n}$ elements and costs $\frac{1}{\sqrt{n}-1}$. Second, we note that one may be concerned that in the first case, the actual γ of the instance is not n. This can be easily remedied as follows. Let the instance consist of $2n$ elements: n elements in the affine plane as in the proof, and another n dummy elements. The dummy elements will be all covered by a single set that arrives first in the input sequence. The remainder of the input sequence is as in the proof. This allows us to argue that the *actual* γ is indeed n, whatever the ensuing scenario is, while decreasing the lower bound by no more than a constant factor.

References

1. Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., Naor, J.: The online set cover problem. SIAM Journal on Computing 39(2), 361–370 (2009)
2. Awerbuch, B., Azar, Y., Plotkin, S.A.: Throughput-competitive on-line routing. In: Proc. 34th FOCS, pp. 32–40 (1993)
3. Bateni, M., Hajiaghayi, M., Zadimoghaddam, M.: Submodular secretary problem and extensions. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, pp. 39–52. Springer, Heidelberg (2010)

4. Berman, P.: A $d/2$ approximation for maximum weight independent set in d-claw free graphs. Nord. J. Comput. 7(3), 178–184 (2000)
5. Buchbinder, N., Naor, J.: Online primal-dual algorithms for covering and packing. Math. Oper. Res. 34(2), 270–286 (2009)
6. Chekuri, C., Khanna, S.: On multidimensional packing problems. SIAM Journal on Computing 33(4), 837–851 (2004)
7. Cygan, M.: Improved approximation for 3-dimensional matching via bounded pathwidth local search. arXiv report 1304.1424 (April 2013)
8. Emek, Y., Halldórsson, M.M., Mansour, Y., Patt-Shamir, B., Radhakrishnan, J., Rawitz, D.: Online set packing. SIAM Journal on Computing 41(4), 728–746 (2012)
9. Feldman, M., Naor, J(S.), Schwartz, R.: Improved competitive ratios for submodular secretary problems (Extended abstract). In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) APPROX/RANDOM 2011. LNCS, vol. 6845, pp. 218–229. Springer, Heidelberg (2011)
10. Freeman, P.: The secretary problem and its extensions: a review. Internat. Statist. Rev. 51(2), 189–206 (1983)
11. Frieze, A.M., Clarke, M.R.B.: Approximation algorithms for the m-dimensional $0-1$ knapsack problem: worst-case and probabilistic analyses. Eur. J. Oper. Res. 15, 100–109 (1984)
12. Gilbert, J.P., Mosteller, F.: Recognizing the maximum of a sequence. J. Amer. Statist. Assoc. 61, 35–73 (1966)
13. Halldórsson, M.M., Kratochvíl, J., Telle, J.A.: Independent sets with domination constraints. Discrete Applied Mathematics 99(1-3), 39–54 (2000)
14. Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Mathematica 182(1), 105–142 (1999)
15. Hazan, E., Safra, S., Schwartz, O.: On the complexity of approximating k-set packing. Computational Complexity 15(1), 20–39 (2006)
16. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. Journal of the ACM 22(4), 463–468 (1975)
17. Magazine, M.J., Chern, M.-S.: A note on approximation schemes for multidimensional knapsack problems. Math. Oper. Res. 9(2), 244–247 (1984)
18. Mansour, Y., Patt-Shamir, B., Rawitz, D.: Overflow management with multipart packets. In: IEEE INFOCOM (2011)
19. Raghavan, P., Thompson, C.D.: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. Combinatorica 7(4), 365–374 (1987)
20. Sahni, S.: Approximate algorithms for the 0/1 knapsack problem. Journal of the ACM 22(1), 115–124 (1975)
21. Srinivasan, A.: Improved approximations of packing and covering problems. In: 27th Annual ACM Symposium on the Theory of Computing, pp. 268–276 (1995)

Multiple Traveling Salesmen in Asymmetric Metrics

Zachary Friggstad*

University of Waterloo, Department of Combinatorics and Optimization, Waterloo, ON, Canada
zfriggstad@math.uwaterloo.ca

Abstract. We consider some generalizations of the Asymmetric Traveling Salesman Path Problem. In these variants, we have multiple salesmen that we are to move around a metric and the goal is to have each node visited by at least one salesman. This should be done while minimizing the total distance travelled by all salesmen.

In the first variant, we are given two nodes s, t and an integer k and the goal is to find k paths from s to t whose union covers all nodes. We give an efficient bicriteria approximation for this problem that uses at most $k + k/b$ paths of total length at most $O(b \log |V|)$ times the optimum value of a natural LP relaxation. By setting b appropriately, we can obtain a true $O(k \log n)$-approximation, an $O(\log n)$-bicriteria approximation using at most $2k$ paths, or, more generally, an $O(\frac{1}{\epsilon} \log n)$-bicriteria approximation using at most $(1 + \epsilon)k$ paths. Prior to this work, only an $O(k^2 \log n)$-approximation and an $O(\log n)$-bicriteria approximation using at most $O(k \log n)$ paths were known.

Next, we consider the case where we have k pairs of nodes $\{(s_i, t_i)\}_{i=1}^{k}$. The goal is to find an $s_i - t_i$ path for every pair such that each node of G lies on at least one of these paths. Simple approximation algorithms are presented for the special cases where the metric is symmetric or where $s_i = t_i$ for each i. We also show that the problem can be approximated within a factor $O(\log n)$ when $k = 2$. On the other hand, we demonstrate that the general problem cannot be approximated within any finite ratio unless P = NP.

Keywords: Traveling Salesman Problem, Linear Programming.

1 Introduction

In the classic form of the Traveling Salesman Problem (TSP), the goal is to find the shortest Hamiltonian cycle in a symmetric metric. The idea is that this describes the fastest way a salesman can visit a given set of clients and then return to their starting position. Christofides' famous algorithm efficiently finds a Hamiltonian cycle of length at most $\frac{3}{2}$ times the optimum solution length [7].

* Research supported by an iCORE ICT/AITF award while studying at the University of Alberta.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 173–188, 2013.

Unfortunately, it is NP-hard to efficiently approximate the optimum solution within any constant better than $\frac{123}{122}$ [16].

Variants of this standard formulation are also natural to consider. For example, the salesman may be required to start and end at given locations, the objective might instead be to minimize the average time a location waits before it is visited by the tour, we might have multiple salesmen at our disposal, etc. Many such variants of TSP in symmetric metrics have constant upper and lower bounds on their approximability. However, the approximability of TSP in asymmetric metrics is not as well understood (see Section 1.1 for some examples). We consider problems of this sort; coordinating the movements of multiple salesmen in an asymmetric metric to ensure each location is visited at least once.

An *asymmetric metric* is a complete directed graph $G = (V, A)$ where V is the set of nodes/locations having nonnegative distances/costs d_{uv} for arcs $uv \in A$. These distances satisfy the directed triangle inequality $d_{uv} \leq d_{uw} + d_{wv}$ for any three distinct nodes $u, v, w \in V$. However, in general it may be that $d_{uv} \neq d_{vu}$ for some nodes $u, v \in V$. If $d_{uv} = d_{vu}$ for every pair $u, v \in V$ then we say that the metric is a *symmetric metric* and we view G as a complete undirected graph. It will sometimes be convenient to say $d_{vv} = 0$ for every $v \in V$ even though we do not have any loops in G. Throughout, we will let n denote the number of nodes in G.

The two main problems we consider are defined as follows. First, in the *k-Person Asymmetric Traveling Salesmen Path Problem* (k-ATSPP) we are given an asymmetric metric $G = (V, A)$ and two distinct nodes $s, t \in V$. The goal is to find k paths from s to t of minimum total cost in G such that every node $v \in V$ lies on at at least one of these paths. Next, we define *General k-ATSPP* to be the following generalization. We are given k pairs of nodes $\{(s_i, t_i)\}_{i=1}^{k}$ in an asymmetric metric G and the goal is to find an $s_i - t_i$ path for each i so that each node lies on at least one such path. This should be done while minimizing the total cost of these paths. The case $k = 1$ for both problems is simply the well-studied Asymmetric Traveling Salesman Path Problem (ATSPP).

What makes k-ATSPP an attractive variant of ATSPP is that the gap between optimum solutions for different values of k in an asymmetric metric may be arbitrarily large. For example, the instance in Figure 1 has a solution of cost 0 using $k = 2$ paths but any single path has cost at least 1. We do not have this large gap in symmetric metrics because a single salesman can cover all k paths by traveling back and forth along these paths between s and t and cover all locations with no greater distance than the total distance of of all k paths (if k is even then one final step from s to t makes this an $s - t$ path while inflating the total distance by at most a $(1 + \frac{1}{k})$-factor).

1.1 Related Work

As mentioned earlier, Christofides' algorithm [7] is a polynomial-time approximation algorithm for classic TSP that finds a Hamiltonian cycle with cost at most $\frac{3}{2}$ times the cost of the optimum solution. Despite the numerous recent advances made on variants and special cases of TSP, no better approximations are

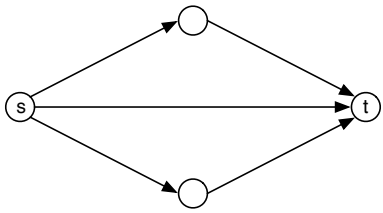


Fig. 1. The pictured arcs all have cost 0 and the missing arcs have cost 1. The gap between optimum solutions for $k = 1$ and $k = 2$ is unbounded.

known for TSP. Hogeveen [14] adapted this approach to TSP Path and obtained a $\frac{3}{2}$-approximation if at most one endpoint is specified and a $\frac{5}{3}$-approximation if both endpoints are specified. In the case that both endpoints are specified, An, Kleinberg, and Shmoys recently described a $\frac{1+\sqrt{5}}{2} < 1.6181$-approximation [1] which was subsequently improved by Sebő to a $\frac{8}{5} = 1.6$-approximation [22]. Currently, the best lower bound on approximating these problems is provided by Karpinski, Lampis, and Schmied who show that there is no c-approximation for any constant $c < \frac{123}{122}$ unless P = NP [16].

In asymmetric metrics, Frieze, Galbiati, and Maffioli [9] gave the first approximation algorithm for ATSP with an approximation ratio of $\log_2 n$ where $n = |V|$. A series of papers improved on this ratio by constant factors [4,15,8] with the last being $\frac{2}{3}\log_2 n$. Finally, Asadpour et al. [2] produced an asymptotically better approximation algorithm for ATSP with ratio $O(\log n / \log\log n)$. Karpinski, Lampis, and Schmied have also shown that there is no c-approximation for these problems for any constant $c < \frac{75}{74}$ unless P = NP [16]. Determining if ATSP has a constant-factor approximation is also a very important open problem.

The variant of finding Hamiltonian paths in asymmetric metrics, namely ATSPP, has only recently been studied from the perspective of approximation algorithms. The first approximation algorithm was an $O(\sqrt{n})$-approximation by Lam and Newman [17]. Following this, Chekuri and Pal [6] brought the ratio down to $O(\log n)$. Finally, Feige and Singh [8] proved that an α-approximation for ATSP implies a $(2+\epsilon)\cdot\alpha$-approximation for ATSPP for any constant $\epsilon > 0$. Combining their result with the ATSP approximation in [2] yields an $O(\log n / \log\log n)$-approximation for ATSPP.

There is a linear programming (LP) relaxation for each of these problems based on the Held-Karp relaxation for TSP [13]. For TSP, this relaxation is

$$\begin{aligned} \text{minimize} : & \textstyle\sum_{uv\in E} d_{uv}x_{uv} \\ \text{such that} : \quad & x(\delta(\{v\})) = 2 \qquad \forall\, v \in V \\ & x(\delta(S)) \geq 2 \qquad \forall\, \emptyset \subsetneq S \subsetneq V \\ & x_{uv} \in [0,1] \;\forall\, uv \in E \end{aligned}$$

where $\delta(S)$ denotes the edges with exactly one endpoint in S and $x(F) = \sum_{uv\in F} x_{uv}$ for any subset of edges F.

Many approximation algorithms for other optimization problems require integrality gap bounds for TSP variants. For example, Nagarajan and Ravi use integrality gap bounds for ATSP in their approximations for the Directed Orienteering and Minimum Density Rooted Cycle problems [18], Bateni and Chuzhoy use ATSP integrality gap bounds in their study of the Directed k-Stroll and k-Tours problems [3], and papers by Nagarajan and Ravi [19] and Friggstad, Salavatipour, and Svitkina [11] both rely on integrality gap bounds for ATSPP in their study of the Minimum Directed Latency problem. While stated slightly differently in [11], the latter also requires a bicriteria approximation for k-ATSPP whose cost can be compared to the optimum solution to LP (1) (see Section 2.1) for the case $k = 2$. We do not attempt to list the numerous applications of integrality gap bounds for symmetric TSP. One recent example is the work of Chakrabarty and Swamy on LP-based formulations for the Minimum Latency Problem in [5].

Many, but not all, of the approximations known for TSP variants also bound the integrality gaps of their respective LP relaxations. For TSP, Wolsey [24] proved the solutions found by Christofides' algorithm [7] are within 3/2 of the optimal solution to the above LP relaxation. For ATSP, Williamson [23] proved that the algorithm of Frieze et al. [9] bounds the integrality gap of the corresponding Held-Karp LP by $\log_2 n$. The improved $O(\log n / \log\log n)$-approximation for ATSP in [2] improved the bound on gap to the same ratio. Sebő's 1.6-approximation for TSP Path also bounds the integrality gap of its relaxation by the same factor [22].

On the other hand, it is not clear that one can adapt the analysis of the first $O(\log n)$-approximation for ATSPP by Chekuri and Pál [6] to an integrality gap bound nor is it clear that the results in [8] relating the approximability of ATSP and ATSPP extend to their integrality gaps. The first integrality gap bound demonstrated for ATSPP was $O(\sqrt{n})$ by Nagarajan and Ravi [19]. This bound was improved to $O(\log n)$ and then to $O(\log n / \log\log n)$ in [11] and [10], respectively.

In the full version of [11], the authors studied extensions of their $O(\log n)$-approximation for ATSPP to k-ATSPP. They demonstrated that k-ATSPP can be approximated within $O(k^2 \log n)$ and that this bounds the integrality gap of LP (1) by the same factor. While not explicitly mentioned in [11], it is easy to get an $O(\log n)$-bicriteria approximation for k-ATSPP that uses at most $O(k \log n)$ paths using their techniques. As far as we know, nothing is known for General k-ATSPP even for the case $k = 2$.

One other problem related to the problems considered in this paper is the following. We are given $2k$ distinct nodes $S = \{s_1, \ldots, s_k\}$ and $T = \{t_1, \ldots, t_k\}$ in a symmetric metric. We want to find k paths whose union spans all nodes. This should be such that each node in S is the start node of exacly one path and each node in T is the end node of exactly one path. Matroid intersection techniques used by Rathinam and Sengupta [20] can be easily adapted to get a 2-approximation for this problem.

1.2 Our Results

By the directed triangle inequality, it is easy to see that there is an optimum solution for an instance of k-ATSPP where each node in $V-\{s,t\}$ lies on precisely one of the k paths. Such an optimum solution corresponds to an integer point in LP (1) (found in Section 2.1) of the same cost. So the optimum value of LP (1), say OPT_{LP}, is a lower bound for the minimum cost k-ATSPP solution. Our main result for k-ATSPP is the following.

Theorem 1. *For any integer $b \geq 1$, there is an efficient algorithm for k-ATSPP that finds between k and $k + \frac{k}{b}$ paths of total cost at most $O(b \log n) \cdot OPT_{LP}$.*

This is a true $O(k \log n)$-approximation when $b = k + 1$ and an $O(\log n)$-approximation using at most $2k$ paths when $b = 1$. More generally, setting $b = \frac{1}{\epsilon}$ results in an $O(\frac{1}{\epsilon} \log n)$-approximation using at most $(1 + \epsilon)k$ paths. The algorithm is also easy to implement with the most complicated subroutine being that of finding a minimum weight perfect matching in a bipartite graph. The running time is dominated by $O(b \log n)$ calls to the matching subroutine. In particular, we are not required to solve LP (1) and we only mention that the approximation guarantee can be stated with respect to OPT_{LP}, and not simply OPT (the optimum k-ATSPP solution cost), to broaden potential applications of this work.

We proceed to study variants of k-ATSPP that vary how the start and/or end locations are specified. Examples are when the start locations are not fixed or when we have a set of k start nodes S and a set of k end nodes T and the start and end locations of the paths should establish a bijection between S and T. Our approximation algorithm for k-ATSPP easily extends to these variants.

Finally, we consider numerous aspects of General k-ATSPP. Our first result is the following.

Theorem 2. *There is an $O(\log n)$-approximation for General 2-ATSPP.*

We also have a 3-approximation for General k-ATSPP in symmetric metrics and an $O(\log n)$-approximation for General k-ATSPP when $s_i = t_i$ for all i. However, it turns out that General k-ATSPP is quite hard if one does not assume any further restrictions.

Theorem 3. *It is NP-hard to distinguish between instances of General k-ATSPP whose optimum solution has cost 0 and instances whose optimum solution has cost at least 1.*

This implies the problem cannot be efficiently approximated within any finite ratio unless P = NP. While the reduction uses $k = n/4$ different (s_i, t_i) pairs, modifications can be made to prove similar hardness results (under assumptions stronger than P $\neq$ NP) with k being as small as polylogarithmic in n.

To summarize, Section 2 presents the algorithm for k-ATSPP, proves Theorem 1, and discusses some variations of k-ATSPP on how the start and/or end locations are specified. In Section 3 we demonstrate an $O(\log n)$-approximation for General 2-ATSPP, discuss approximation algorithms for other restrictions of General k-ATSPP, and prove Theorem 3. Section 4 then concludes this paper and identifies some directions for future work.

2 A Bicriteria Approximation for k-ATSPP

In this section, we will develop a bicriteria approximation algorithm that finds approximately k paths from s to t in an asymmetric metric $G = (V, A)$ whose total cost is within some bounded ratio of the optimum value of LP relaxation (1). The algorithm is parameterized by a positive integer b; different bicriteria approximation guarantees result from different choices of b.

2.1 Preliminaries

If X is a flow between two nodes or a circulation then we let X_{uv} denote the value that X assignes to arc $uv \in A$. For $S \subseteq V$ we let $X(\delta^+(S)) = \sum_{u \in S, v \in V-S} X_{uv}$ and $X(\delta^-(S)) = \sum_{v \in V-S, u \in S} X_{vu}$. For brevity, we let $X(\delta^+(v)) := X(\delta^+(\{v\}))$ and $X(\delta^-(v)) := X(\delta^-(\{v\}))$ for $v \in V$. When the underlying graph may not be clear from the context, we use the notation δ_G to indicate that the set of edges considered in the cut come from graph G. We say X is integral if X_{uv} is an integer for each arc $uv \in A$. The cost of X is $\sum_{uv \in A} d_{uv} \cdot X_{uv}$. All flows and circulations X in this paper will have $X_{uv} \geq 0$ for each arc $uv \in A$. The value of an $s-t$ flow F is $F(\delta^+(s)) - F(\delta^-(s))$, the amount of flow sent from s to t. We say that an $s-t$ flow F is supported by a node $v \neq s, t$ if $F(\delta^+(v)) > 0$.

Our starting point will be to use structures similar to cycle covers from [9] and path/cycle covers from [17] and [11].

Definition 1. *A k-path/cycle cover from s to t is an integral $s-t$ flow F such that $F(\delta^+(v)) = F(\delta^-(v)) = 1$ for each $v \in V - \{s, t\}$, $F(\delta^+(s)) = F(\delta^-(t)) = k$, and $F(\delta^-(s)) = F(\delta^+(t)) = 0$.*

Note that in a k-path/cycle cover F the flow F_{uv} across any arc $uv \in A - \{st\}$ is either 0 or 1 and $F_{st} \leq k$. If we regard F as a multiset of arcs, then F may be decomposed into k paths from s to t and a collection of cycles where every $v \in V - \{s, t\}$ lies on exacly one path or exactly one cycle. We can efficiently find a minimum-cost k-path/cycle cover using a simple reduction to minimum weight perfect matching in a bipartite graph with $n + k - 2$ nodes on each side.

LP (1) is the LP relaxation for k-ATSPP we consider. It is similar to the LP relaxation for ATSPP considered in [10,11,19].

$$\begin{aligned}
\text{minimize :} \quad & \sum_{e \in A} d_{uv} x_{uv} && && (1)\\
\text{subject to :} \quad & x(\delta^+(v)) = x(\delta^-(v) = 1 && \forall v \in V - \{s, t\} &\\
& x(\delta^+(s)) = x(\delta^-(t)) = k && &\\
& x(\delta^-(s)) = x(\delta^+(t)) = 0 && &\\
& x(\delta^+(S)) \geq 1 && \forall \{s\} \subseteq S \subsetneq V & (2)\\
& x_{uv} \geq 0 && \forall\ uv \in A &
\end{aligned}$$

2.2 The Algorithm

Let $b \geq 1$ be an integer, this is the b in the statement of Theorem 1. For notational convenience, we will let L be $(b+1)\lfloor \log_2 n \rfloor$ for the remainder of this section.

The algorithm consists of two phases. The first phase is identical to the first phase of the algorithm in [11] that repeatedly finds k-path/cycle covers and discards some carefully chosen nodes in each iteration. Again, we emphasize that while we are comparing the cost of the solution to the optimum solution value of LP (1), the algorithm itself is purely combinatorial and does not need to explicitly solve the LP. The following follows from work in [11].

Lemma 1. *There is an efficient algorithm that finds a subset of nodes W containing s and t, an integral $s-t$ flow F of value $k \cdot L$ that is not supported by any $v \in V - W$, and an integral circulation C such that:*

1. *The support of F is acyclic.*
2. *$F(\delta^+(v)) = F(\delta^-(v)) \geq L - \lfloor \log_2 n \rfloor = b\lfloor \log_2 n \rfloor$ for each $v \in W - \{s,t\}$.*
3. *$C(\delta^+(v)) = C(\delta^-(v)) \geq 1$ for each $v \in V - W$.*
4. *Every strongly connected component in the support of C has a node in W.*
5. *$F + C$ has cost at most $L \cdot OPT_{LP}$.*

Viewing F as a collection of $k \cdot L$ paths, we can immediately get a bicriteria k-ATSPP approximation with similar guarantees as the algorithm in [11] in the following way. Since each component of C is already Eulerian, we can transform it to a Hamiltonian cycle on the same set of nodes of no greater cost. Since each component of C is "anchored" in some node covered by F then these cycles can be grafted into the paths of F. The total cost of this solution would be at most $L \cdot OPT_{LP}$ which provides an $O(b \log n)$-approximation using at most $L \cdot k = O(bk \log n)$ paths. Using $b = 1$ recovers the result from [11].

However, Theorem 1 states that the number of paths used in the solution *decreases* as b increases. In particular, it promises that at most $k + \frac{k}{b}$ are used. This is accomplished by the second phase of our algorithm. The main goal of this phase is to identify an integral flow F' of value at most $k + \frac{k}{b}$ such that $F' \leq F$ (on an arc-by-arc basis) such that F' is supported by each $v \in W - \{s,t\}$. The key observation here is that each $v \in W$ supports a lot of flow in F (part 2 of Lemma 1), so scaling F by a large amount results in a (fractional) flow of much smaller value that still passes through each $v \in W$ to an extent of 1. Using integrality of flows plus the fact that F is acyclic, we can find an integral flow of no greater cost that is still supported by each $v \in W$. Then we can graft the circulations of C into these at most $k + \frac{k}{b}$ paths in the manner described above. The details of this procedure are described below.

2.3 The Second Phase

Consider the acyclic integral flow F found by the first phase (cf. Lemma 1). The main object of concern in this step is the following polytope $\mathcal{P}(D)$ where $D \in \mathbb{R}$.

In $\mathcal{P}(D)$, we have a variable z_{uv} for every arc uv in the subgraph $G[W]$ of the asymmetric metric G induced by W. The full description of $\mathcal{P}(D)$ is:

$$\begin{aligned} z(\delta^+(w)) = z(\delta^-(w)) \geq \;& 1 && \forall w \in W - \{s,t\} && (3)\\ z(\delta^+(s)) = z(\delta^-(t)) = \;& D && && (4)\\ z(\delta^-(s)) = z(\delta^+(t)) = \;& 0 && && (5)\\ 0 \leq z_{uv} \leq \;& F_{uv} && \forall \text{ arcs } u,v \text{ in } G[W] && (6) \end{aligned}$$

Since the support of F is acyclic and the support of z is required to be a subset of the support of F, then any integral point $z \in \mathcal{P}(D)$ corresponds to an integral flow P of cost at most the cost of F such that a path decomposition of P yields a collection of D paths from s to t whose union covers all nodes in W. Thus, our goal is to find a value $D \in [k, k + \frac{k}{b}]$ for which $\mathcal{P}(D)$ has an integer point. When D is an integer, $\mathcal{P}(D)$ describes an $s-t$ flow with integer upper and lower bounds on the amount of flow across each edge and the amount of flow through each vertex. So, the following holds because of total unimodularity (eg. [21]).

Lemma 2. *Every extreme point of polytope $\mathcal{P}(D)$ is integral when D is an integer.*

Thus, to prove $\mathcal{P}(D)$ has an integer point for some integer D it suffices to prove that $\mathcal{P}(D)$ contains *any* point. That is, for $D \in \mathbb{Z}$ if there is some point $z \in \mathcal{P}(D)$ with, perhaps, rational entries, then there is also an integral point z'.

The following lemma is the first step to finding a good integer D for which $\mathcal{P}(D) \neq \emptyset$.

Lemma 3. $\mathcal{P}\left(\frac{kL}{b\lfloor \log_2 n \rfloor}\right) \neq \emptyset$.

Proof. Define a point z by $z_{uv} = \frac{F_{uv}}{b\lfloor \log_2 n \rfloor}$ for every arc uv in $G[W]$. Then Lemma 1 implies $z(\delta^+(w)) = z(\delta^-(w)) = \frac{F(\delta^+(w))}{b\lfloor \log_2 n \rfloor} \geq 1$ for each $w \in W - \{s,t\}$ so Constraint (3) holds. Similarly, Constraints (4) and (5) hold because $F(\delta^+(s)) = F(\delta^-(t)) = kL$ and $F(\delta^-(s)) = F(\delta^+(t)) = 0$. Finally, $b \geq 1$ means $b\lfloor \log_2 n \rfloor \geq 1$ so we have the component-wise domination $z \leq F$.

Note that the $\frac{kL}{b\lfloor \log_2 n \rfloor}$ may not be an integer. This is easily remedied.

Lemma 4. *If $\mathcal{P}(D) \neq \emptyset$, then $\mathcal{P}(\lfloor D \rfloor) \neq \emptyset$.*

Proof. If D is an integer, then there is nothing to show. Otherwise, let z^* be any point in $\mathcal{P}(D)$. Consider the variant $\mathcal{P}'(D)$ of $\mathcal{P}(D)$ obtained by relaxing Constraints (4) to

$$\lfloor D \rfloor \leq z(\delta^+(s)) = z(\delta^-(t)) \leq \lceil D \rceil.$$

Note that $z^* \in \mathcal{P}'(D)$ as $\mathcal{P}(D) \subseteq \mathcal{P}'(D)$.

Since $\mathcal{P}'(D)$ describes an $s-t$ flow with integer upper and lower bounds, then by total unimodularity (eg. [21]) z^* can be decomposed as a convex combination of integer flows in $\mathcal{P}'(D)$. Since $z^*(\delta^+(s)) = D < \lceil D \rceil$ (as D is not an integer), then this convex combination supports a point z' with $z'(\delta^+(s)) = \lfloor D \rfloor$. In particular, $z' \in \mathcal{P}(\lfloor D \rfloor)$.

Corollary 1. *There is an integer $k' \in [k, k + \frac{k}{b}]$ such that $\mathcal{P}(k') \neq \emptyset$. That is, we can efficiently find between k and $k + \frac{k}{b}$ paths from s to t whose union spans all nodes in W. Furthermore, the cost of these paths is at most the cost of the flow F constructed in the first phase.*

Proof. We just proved $\mathcal{P}\left(\left\lfloor \frac{kL}{b\lfloor \log_2 n \rfloor} \right\rfloor\right) \neq \emptyset$ and an integral point can be efficiently found in this LP because we can efficiently find an extreme point of the polytope (flow-based techniques can also be used). The result follows since

$$k \leq \left\lfloor \frac{kL}{b\lfloor \log_2 n \rfloor} \right\rfloor \leq \frac{k(b+1)\lfloor \log_2 n \rfloor}{b\lfloor \log_2 n \rfloor} = k + \frac{k}{b}.$$

To complete the proof of Theorem 1, we form a circulation C' in the following way. View the paths from Corollary 1 as an integral flow of value k', add the circulation C from Lemma 1, and set the value of the arc ts in C' to k'.

Lemma 5. *C' is an integral circulation whose support is strongly connected in $G(V, A)$.*

So, when viewing C' as a multiset of arcs, we see the resulting graph is Eulerian whose total arc cost is at most $L \cdot OPT_{LP}$ plus the cost of the k' arcs form t to s. Follow an Eulerian circuit and remove the k' different $t - s$ arcs to get k' walks from s to t whose union spans all nodes. By triangle inequality, these walks can be shortcut to paths of cost at most $L \cdot OPT_{LP}$ so that each node in $V - \{s, t\}$ lies on exactly one such path.

2.4 Varying the Endpoints

Consider the following different ways to specify the start locations of the paths of a k-ATSPP instance: each path may start at any node (No Source), all paths start at a common node s (Common Source), or there are nodes $s_1, \ldots, s_k$ where each must be the start of some path (Multiple Sources). We can also consider analogous ways to specify the end locations of the paths. In Multiple Sources, Multiple Sinks instances, we only require each path start at some s_i and end at some t_j. It may be that some paths start and end at locations with different indices.

The following theorems are easy to verify and the proofs are only briefly sketched. We let OPT be the cost of the optimum solution using exactly k paths for the k-ATSPP variant in question.

Theorem 4. *For any integer $b \geq 1$, there is an algorithm for the No Source, Single Sink variant of k-ATSPP that finds between k and $k + \frac{k}{b}$ paths of total cost at most $O(b \log n) \cdot OPT$.*

Proof. Simply add a new start node s and set $sv = 0$ and $vs = \infty$ for every $v \in V$. Then use the approximation algorithm from Theorem 1.

Theorem 5. *For any integer $b \geq 1$, there is an algorithm for the Multiple Sources, Single Sink variant of k-ATSPP that finds between k and $k + \frac{k}{b}$ paths of total cost at most $O(b \log(n + k)) \cdot OPT$.*

Proof. Let $s_1, \ldots, s_k$ be the multiple sources. Create a start node s and k other new nodes $s'_1, \ldots, s'_k$. Add cost 0 arcs from s to s'_i and from s'_i to s_i for each i. Add a cost ∞ arc from v to s for every $v \in V$. Use Theorem 1 on the shortest paths metric of this graph. The intermediate nodes s'_i are to ensure that each s_i is the start location of some path.

Combining the constructions in Theorems 4 and 5 can also be used to combine different start and end location specifications (eg. No Source, Multiple Sinks).

3 General k-ATSPP

Recall that in General k-ATSPP, we are given k pairs of nodes $(s_1, t_1), \ldots, (s_k, t_k)$. The goal is to find an $s_i - t_i$ path for each $1 \leq i \leq k$ so that every $v \in V$ lies on at least one such path. This differs from the Multiple Sources, Multiple Sinks variant described in Section 2.4 since the path at s_i must end at t_i, rather than merely requiring that the start and end nodes of the paths establish a bijection between $\{s_1, \ldots, s_k\}$ and $\{t_1, \ldots, t_k\}$.

3.1 Approximating General 2-ATSPP

Let (s_1, t_1) and (s_2, t_2) be pairs of nodes we are to connect. Furthermore, suppose all four of these endpoints are distinct (by creating multiple copies of locations if necessary). This means there is an optimum General 2-ATSPP solution where the two paths are vertex disjoint.

Let P'_1 and P'_2 be $s_1 - t_1$ and $s_2 - t_2$ paths, respectively, in a fixed optimum solution of cost OPT. For two nodes u, v we write $u \prec v$ if both u and v appear on a common path P'_1 or P'_2 and u appears earlier than v on this path.

Notice that the optimum Multiple Sources, Multiple Sinks 2-ATSPP solution for the case with sources $\{s_1, s_2\}$ and sinks $\{t_1, t_2\}$ is a lower bound for OPT. The first step of the algorithm is to run an α-approximation for this variant of 2-ATSPP. This gives us two paths P_1, P_2 starting at s_1, s_2, respectively. If P_1 ends at t_1 (equivalently, P_2 ends at t_2), then these paths form a valid solution for the General 2-ATSPP problem with cost at most $\alpha \cdot OPT$. Otherwise, we use the following lemma which implies a cheap and efficient way to modify these paths to get a feasible General k-ATSPP solution.

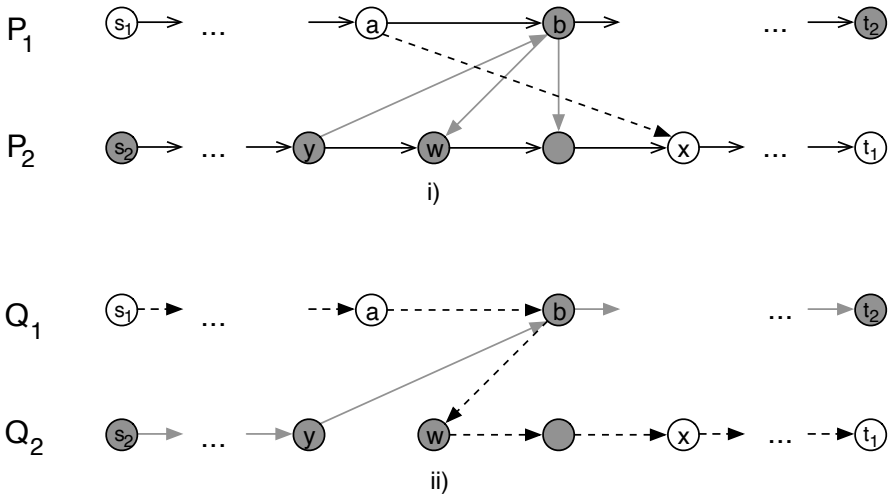


Fig. 2. i) An illustration of the case $y \in P_2'$ with immediate successor $w \neq x$ on P_2. The white nodes lie on P_1' and the gray nodes lie on P_2'. A dashed arc $v \to v'$ indicates $v \prec v'$ on P_1' and a gray arc $v \to v'$ indicates $v \prec v'$ on P_2'. ii) The $s_1 - t_1$ and $s_2 - t_2$ paths Q_1, Q_2 formed in the proof of Lemma 7 for this case.

Lemma 6. *There are nodes u_1, v_2 on P_1 and v_1, u_2 on P_2 such that the following hold: a) $u_1 = v_2$ or u_1v_2 is an arc on P_1, b) $v_1 = u_2$ or v_1u_2 is an arc on P_2 and c) $d_{u_1u_2} + d_{v_1v_2} \leq OPT$.*

Proof. Let a, b be such that a is on P_1', b is on P_2' and a appears immediately before b on P_1. Such nodes exist because P_1 begins with $s_1 \in P_1'$ and ends in $t_2 \in P_2'$ so at some point along P_1 there is an arc beginning in P_1' and ending in P_2'. If there are multiple such points a, b, then choose the one where a appears earliest on P_1.

Let x be the first node along P_2 such that $a \prec x$. We know such a node exists because $a \prec t_1$ and $t_1 \in P_2$. Now, let y be the furthest node along P_2 but still before x such that either $y \in P_1'$ or $y \in P_2'$ and $y \prec b$. Again, such a node exists because $s_2 \in P_2'$ and $s_2 \prec b$.

Suppose $y \in P_2'$ with $y \prec b$. If y is immediately followed by x on P_2 then we let $u_1 = a, u_2 = x, v_1 = y, v_2 = b$. Otherwise, say w is the immediate successor of y on P_2. By our choice of y we have $w \in P_2'$ and $b \prec w$. In this case, we set $u_1 = b, u_2 = w, v_1 = y, v_2 = b$. This case is illustrated in Figure 2.

Next, suppose $y \in P_1'$. Let z be the first node on P_2 that lies on P_1'. Note that z occurs no later than y on P_2 (it may be equal to y). Now, by our choice of a on P_1, every node between s_1 and a on P_1 also lies on P_1'. We also have that s_1 appears before z on P_1' (since s_1 is the start of P_1') and, by our choice of x and the fact that z appears before x on P_2, we have that z appears before a on P_1'. So, there must be some arc cd on the subpath of P_1 starting at s_1 and ending at a such that c appears before z on P_1' and d appears after z on P_1'. We let $u_1 = c, u_2 = z, v_1 = z, v_2 = d$ in this case.

In all cases, we can easily verify that conditions a) and b) in the statement of the lemma are satisfied. Also, in any case we have that one of the following is true:

1. $u_1 \prec u_2$ and $v_1 \prec v_2$ with u_1 and v_1 appearing on different paths in P'_1, P'_2
2. $v_1 \prec v_2 = u_1 \prec u_2$
3. $u_1 \prec u_2 = v_1 \prec v_2$

For any of these, it is easy to see that the triangle inequality implies $d_{u_1u_2} + d_{v_1v_2}$ is a lower bound on the total cost of P'_1 and P'_2. That is, $d_{u_1u_2} + d_{v_1v_2} \leq OPT$.

Lemma 7. *If there is an α-approximation for the Multiple Sources, Multiple Sinks variant of 2-ATSPP then there is an $(\alpha + 1)$-approximation for General 2-ATSPP.*

Proof. As mentioned before, we use the α-approximation for the Multiple Sources, Multiple Sinks variant of 2-ATSPP with starting nodes $\{s_1, s_2\}$ and ending nodes $\{t_1, t_2\}$. Say P_1 and P_2 are the two paths starting at s_1 and s_2, respectively. If P_1 also ends at t_1 then P_2 ends at t_2 and we are done.

Otherwise, we proceed as follows. Try all $O(n^2)$ guesses for u_1, u_2 on P_1 and v_1, v_2 on P_2 where either $u_1 = v_2$ or u_1v_2 is an arc on P_1 and either $v_1 = u_2$ or v_1u_2 is an arc on P_2. For each guess, construct a path Q_1 by traveling along P_1 from s_1 to u_1, then using the u_1u_2 arc in G, and then traveling along P_2 from u_2 to t_1. Similarly construct Q_2 by traveling from s_2 to v_1 on P_2, then using the v_1v_2 arc in G, and then traveling along P_1 from v_2 to t_2.

It is easy to see that Q_1 and Q_2 form a feasible solution for this General 2-ATSPP instance. Since each arc on P_1 and P_2 is traversed at most once by Q_1 and Q_2, then the total cost of Q_1 and Q_2 is at most $\alpha \cdot OPT + d_{u_1u_2} + d_{v_1v_2}$. Output the cheapest solution found over these guesses. When the algorithm guesses nodes u_1, u_2, v_1, v_2 from Lemma 6 we have $d_{u_1u_2} + d_{v_1v_2} \leq OPT$. So the final cost of Q_1 and Q_2 is at most $(\alpha + 1) \cdot OPT$.

By setting $b = 3$ and using Theorem 5 composed with an analogous result for Multiple Sinks instances, we get an $O(\log n)$-approximation for the Multiple Sources, Multiple Sinks variant of 2-ATSPP. Combining this approximation with Lemma 7 proves Theorem 2.

3.2 Approximating Other Restrictions of General k-ATSPP

We briefly mention a couple of variants of General k-ATSPP that can be approximated well. We leave their full descriptions to the full version of this paper.

The first variant is when the metric is symmetric. In this case, there is a simple 3-approximation using a tree doubling approach. Next, if $s_i = t_i$ for each $1 \leq i \leq k$, then the problem is to find a cycle cover where each cycle contains some root node s_i where we allow cost 0 loops on the nodes s_i (corresponding to the salesman at s_i going directly to t_i). This version can be approximated within $\lfloor \log_2(n-k) \rfloor + 1$ using a modification of the ATSP algorithm by Frieze et al. [9].

3.3 Hardness of General k-ATSPP

The following NP-complete [12] problem is used in our reduction.

Definition 2. *In the Tripartite Triangle Partition problem, we are given a tripartite graph $G = (U \cup V \cup W, E)$ with $|U| = |V| = |W| = n$ where no edge in E has both endpoints in a common set U, V, or W. The problem is to determine if it is possible to find n vertex-disjoint triangles (size 3 cliques) in G.*

Technically, [12] only proves NP-completeness of the Partition Into Triangles problem where the problem is to determine if a *general* graph can be partitioned into cliques of size 3. However, if we use their reduction starting with a 3D Matching instance (instead of the more general problem Exact Cover by 3-Sets they use), then the resulting graphs are indeed tripartite and that the tripartition is explicitly computed in the reduction so it can be presented as part of the input of the resulting Tripartite Triangle Partition instance.

Let $G = (U \cup V \cup W, E)$ be an instance of Tripartite Triangle Partition with $|U| = |V| = |W| = n$. Create a directed graph H with four layers of nodes X_1, X_2, X_3, X_4 where X_1 and X_4 are disjoint copies of U, X_2 is a copy of V, and X_3 is a copy of W. For every edge e in G, there is a unique index $1 \leq i \leq 3$ such that the endpoints of e lie in X_i and X_{i+1}. Add this arc to H, direct it from X_i to X_{i+1}, and set its cost to 0. This is illustrated in Figure 3.

Set $k := n$ and consider the General k-ATSPP instance on H' obtained from the shortest paths metric H where we set the cost of a uv arc to be 1 if there is no $u - v$ path in H. For each $u \in U$, we have a source/sink pair from the copy of u in X_1 to the copy of u in X_4. The details of the following claim are simple and the proof of Theorem 3 immediately follows.

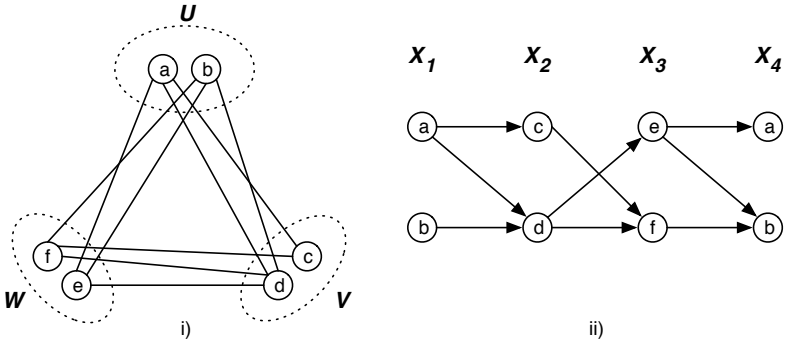


Fig. 3. i) An instance of tripartite triangle partition with $n = 2$. ii) The corresponding graph H.

Claim. There is a Tripartite Triangle Partition solution in G if and only if the optimum General k-ATSPP solution in H' solution has cost 0.

Note that the value k is $n/4$ in the above reduction where we now let n be the number of nodes in the resulting General k-ATSPP instance. Through padding arguments, we can establish the same result with $k = n^{\epsilon}$ for any constant $\epsilon > 0$. Furthermore, under the Exponential Time Hypothesis one can show that there is no polynomial-time, finite-ratio approximation for General k-ATSPP with $k = O(\log^c n)$ for some constant c. The details behind these claims are deferred to the full version of this paper. The complexity of approximating the case when k is a constant at least 3 remains open.

4 Future Directions

Our best approximation for k-ATSPP that uses exactly k paths has an approximation guarantee of $O(k \log n)$. Can the dependence on k in the approximation ratio be reduced? Perhaps there is an $O(\text{polylog}(n, k))$-approximation for k-ATSPP that uses only k paths. On the other hand, the problem might be hard to approximate much better than k. Also, as far as we know the integrality gap of LP (1) could be $\Omega(k)$.

For General k-ATSPP, the case $k = 1$ is simply ATSPP and we described an $O(\log n)$-approximation for $k = 2$. Is there a more general $O(f(k) \cdot \log n)$-approximation for General k-ATSPP whose running time is polynomial when k is a constant?

Finally, rather than minimizing the total cost of all paths we might want to minimize the cost of the most expensive path. This can be thought of as minimizing the time it takes for agents moving simultaneously to visit all locations. From Theorem 1 and the observation that the total cost of k paths is at most k times the cost of the most expensive path, we get an $O(bk \log n)$-approximation for this variant that uses at most $k + \frac{k}{b}$ paths.

Using known algorithms for related vehicle routing problems, we can significantly reduce the approximation ratio at the expense of increasing the number of paths used in the final solution.

Theorem 6. *If there is an α-approximation for the Point-to-Point Directed Orienteering problem, then we can efficiently find $O(k\alpha \log n)$ paths of length at most OPT.*

Theorem 7. *If there is an α-approximation for the Directed k-Stroll problem, then we can efficiently find $O(k \log n)$ paths of length at most $\alpha \cdot OPT$ each.*

In particular, the current best approximation algorithms for Directed Orienteering [18] and Directed k-Stroll [3] imply the following.

Corollary 2. *We can efficiently find $O(k \log^3 n / \log \log n)$ paths where the cost of each path does not exceed OPT.*

Corollary 3. *There is an $O(\log^3 n / \log \log n)$-bicriteria approximation using at most $O(k \log n)$ paths.*

We leave it as an open problem to improve these bounds. In particular, is it possible to obtain a polylogarithmic approximation that uses only $O(k)$ paths? We also note that the hardness results for General k-ATSPP proven in this paper also hold for the the variant where we want to minimize the maximum cost of the $s_i - t_i$ paths.

Acknowledgements. The author would like to thank Anupam Gupta, Mohammad R. Salavatipour, and Zoya Svitkina for insightful discussions on these problems.

References

1. An, H.-C., Kleinberg, R., Shmoys, D.B.: Improving christofides' algorithm for the s-t path TSP. In: Proceedings of ACM Symposium on Theory of Computing, pp. 875–886 (2012)
2. Asadpour, A., Goemans, M.X., Madry, A., Oveis Gharan, S., Saberi, A.: An $O(\log n/\log\log n)$-approximation algorithm for the asymmetric traveling salesman problem. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms, pp. 379–389 (2010)
3. Bateni, M., Chuzhoy, J.: Approximation algorithms for the directed k-tour and k-stroll problems. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, pp. 25–38. Springer, Heidelberg (2010)
4. Blaser, M.: A new approximation algorithm for the asymmetric TSP with triangle inequality. ACM Trans. Algorithms 4(4), 47:1–47:15 (2008)
5. Chakrabarty, D., Swamy, C.: Facility location with client latencies: Linear programming based techniques for minimum latency problems. In: Günlük, O., Woeginger, G.J. (eds.) IPCO 2011. LNCS, vol. 6655, pp. 92–103. Springer, Heidelberg (2011)
6. Chekuri, C., Pál, M.: An $O(\log n)$ approximation ratio for the asymmetric traveling salesman path problem. Theory of Computing 3(1), 197–209 (2007)
7. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University (1976)
8. Feige, U., Singh, M.: Improved approximation ratios for traveling salesperson tours and paths in directed graphs. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) APPROX and RANDOM 2007. LNCS, vol. 4627, pp. 104–118. Springer, Heidelberg (2007)
9. Frieze, A.M., Galbiati, G., Maffioli, F.: On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. Networks 12(1), 23–39 (1982)
10. Friggstad, Z., Gupta, A., Singh, M.: An improved integrality gap for asymmetric TSP paths. In: Goemans, M., Correa, J. (eds.) IPCO 2013. LNCS, vol. 7801, pp. 181–192. Springer, Heidelberg (2013)
11. Friggstad, Z., Salavatipour, M.R., Svitkina, Z.: Asymmetric traveling salesman path and directed latency problems. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, pp. 419–428 (2010); Full version: manuscript number 0907.0726 on arXiv
12. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)

13. Held, M., Karp, R.M.: The Traveling-Salesman Problem and Minimum Spanning Trees. Operations Research 18, 1138–1162 (1970)
14. Hoogeveen, J.A.: Analysis of christofides' heuristic: Some paths are more difficult than cycles. Oper. Res. Lett. 10(5), 291–295 (1991)
15. Kaplan, H., Lewenstein, M., Shafrir, N., Sviridenko, M.: Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. J. ACM 52(4), 602–626 (2005)
16. Karpinski, M., Lampis, M., Schmied, R.: New inapproximability bounds for TSP. CoRR, abs/1303.6437 (2013)
17. Lam, F., Newman, A.: Traveling salesman path problems. Math. Program. 113(1), 39–59 (2008)
18. Nagarajan, V., Ravi, R.: Poly-logarithmic approximation algorithms for directed vehicle routing problems. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) APPROX and RANDOM 2007. LNCS, vol. 4627, pp. 257–270. Springer, Heidelberg (2007)
19. Nagarajan, V., Ravi, R.: The directed minimum latency problem. In: Goel, A., Jansen, K., Rolim, J.D.P., Rubinfeld, R. (eds.) APPROX and RANDOM 2008. LNCS, vol. 5171, pp. 193–206. Springer, Heidelberg (2008)
20. Rathinam, S., Sengupta, R.: Matroid intersection and its application to a multiple depot, multiple TSP. Technical report, Institute of Transportation Studies, UC Berkeley (2006)
21. Schrijver, A.: Combinatorial Optimization - Polyhedra and Efficiency. Springer (2003)
22. Sebő, A.: Eight-fifth approximation for the path TSP. In: Goemans, M., Correa, J. (eds.) IPCO 2013. LNCS, vol. 7801, pp. 362–374. Springer, Heidelberg (2013)
23. Williamson, D.P.: Analysis of the held-karp heuristic for the traveling salesman problem. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA (1990)
24. Wolsey, L.A.: Heuristic analysis, linear programming and branch and bound. In: Combinatorial Optimization II. Mathematical Programming Studies, vol. 13, pp. 121–134. Springer, Heidelberg (1980)

Approximate Indexability and Bandit Problems with Concave Rewards and Delayed Feedback*

Sudipto Guha[1,**] and Kamesh Munagala[2,***]

[1] Department of Computer and Information Sciences, University of Pennsylvania, Philadelphia, PA
sudipto@cis.upenn.edu
[2] Department of Computer Science, Duke University, Durham, NC 27708-0129
kamesh@cs.duke.edu

Abstract. We consider two stochastic multi-armed bandit problems in this paper in the Bayesian setting. In the first problem the accrued reward in a step is a concave function (such as the maximum) of the observed values of the arms played in that step. In the second problem, the observed value from a play of arm i is revealed after δ_i steps. Both of these problems have been considered in the bandit literature but no solutions with provably good performance guarantees are known over short horizons. The two problems are similar in the sense that the reward (for the first) or the available information (for the second) derived from an arm is not a function of just the current play of that arm. This interdependence between arms renders most existing analysis techniques inapplicable.

A fundamental question in regard to optimization in the bandit setting is *indexability*, i.e., the existence of near optimal index policies. Index policies in these contexts correspond to policies over suitable single arm state spaces which are combined into a global policy. They are extremely desirable due to their simplicity and perceived robustness, but standard index policies do not provide any guarantees for these two problems. We construct $O(1)$ approximate (near) index policies in polynomial time for both problems. The analysis identifies a suitable subset of states for each arm such that index policies that focus on only those subsets are $O(1)$-approximate.

1 Introduction

In the Multi-Armed Bandit (MAB) problem an agent decides on allocating resources between competing actions (arms) with uncertain rewards and can only take a small set of actions at a time (play the arms). The agent observes the outcomes (termed as "values") from only the arms which are played – and collects a reward which is a suitable function of the observed values. For the problems we

* This paper subsumes the unpublished manucript [23].
** Research supported by NSF awards CCF-0644119, CCF-1117216 and a gift from Google.
*** Supported by an Alfred P. Sloan Research Fellowship, an award from Cisco, and by NSF grants CCF-0745761, CCF-1008065, and IIS-0964560.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 189–204, 2013.

consider in this paper, the goal of the agent is to maximize the sum of per-step rewards over a horizon of T steps. The task of an algorithm in this context is to provide the agent with a *decision policy* which is a mapping from the current state of the system (comprising the states of all the arms and possibly a few other parameters) to an *action*, such as playing a subset of arms or stopping altogether. Of particular interest are decomposable *index policies*, where each arm is reduced to a single *priority* value based on current state of that arm (and possibly a few global parameters) and the feasible subset of arms with the highest combined priority is played. See [9,8,16] for detailed surveys of the MAB literature.

The problems we consider herein are set apart from most problems in the literature due to the presence of three aspects (i) The number of arms are large in comparison to the optimization horizon (ii) There exist prior information about the reward that can be accrued from each arm and (iii) There exists natural constraints such as budgets. All three of these aspects hold (with varying levels) in context of advertising, content delivery, and route selection, where the arms correspond to advertisers, (possibly machine generated) webpages, and (machine generated) routes respectively.

The first aspect is motivated by the fact that MAB problems are increasingly being used to model situations where the arms (available actions, say n in number) are machine generated and large in comparison to the optimization horizon T. This is in contrast to the historical development of MAB problems which considered few alternatives (such as in medical treatments or hypothesis testing) – the fact that T/n is not overwhelmingly large implies that standard concentration of measure type results do not provide good bounds. Optimization under short horizons necessitate a model, which brings us to the second aspect. Moreover many of the application scenarios for MABs are repetitive, that is, multiple instances are required to be solved. This suggests a Bayesian model where prior information is available and is updated (using Bayes' rule) as the algorithm proceeds. The third aspect of budgets (the total reward that can be derived from any specific alternative) arise naturally in many variants of the applications mentioned above – budgets are the most natural constraints in any optimization and especially in resource allocation problems. We now formally define the Bayesian MAB setting we consider.

Definition 1 (Finite Horizon Bayesian MAB Setting). *We have a set of n independent arms. Arm i provides rewards based on an unknown distribution D_i drawn from a known prior distribution*[1] *$\mathcal{D}_i$. At each step, the decision policy can play a set of arms and observe the outcome of* **only** *the played arms. Based on the outcomes, the priors of the arms that were played in the previous step are updated using Bayes' rule. If arm i was played then one more observation is available and the state of arm i transitions to the appropriate posterior distribution, which acts as the prior for the next play of arm i. The evolution of the arm i is modeled naturally by a DAG $\mathcal{S}_i$ where each vertex $u \in \mathcal{S}_i$ corresponds to a prior distribution. The root corresponds to the initial prior $\mathcal{D}_i$. Every other*

[1] This paper does not assume any specific prior, but assumes that (a) the prior can be updated efficiently and (b) the underlying true object is a distribution.

state $u \in \mathcal{S}_i$ corresponds to a set of observations[2]*, and hence to the posterior X_{iu} obtained by applying Bayes rule to $\mathcal{D}_i$ with the those observations. $\mathcal{S}_i$ is defined to be the state space of the arm i.*

We consider two problems in the Bayesian MAB setting, focusing on concave rewards and delayed feedback.

Problem 1 (**MaxMAB Problem***).* In the MaxMAB problem[3] we make at most K simultaneous plays on different arms at a step and the accrued reward at a step is the maximum of the observations which are available immediately. The goal is to maximize the expected sum of the rewards over the T time steps, where the expectation is taken over the priors as well as the outcome of each play. While several variants can be defined based on budgets (the total reward that can be accrued from any arm), the most interesting variant corresponds to the case where only the budget of the arm contributing to the reward is depleted.

Such problems arise in fault-tolerant computation – for example, the same packet may be sent on K independent routes [2], but the reward is no greater if multiple copies of the packet are received. This also models an Ad-exchange where on seeing an opportunity to show an ad, the exchange solicits bids from K different advertisers (ad-networks) and the highest bidder is allocated the ad and charged their bid. The losing bidders incur no charge.

*Problem 2 (***Delayed Feedback).** In this problem the observed reward from a play of arm i is obtained after some δ_i time steps. For simplicity we assume that a single arm is played at a time step. The arms have budgets which can be incorporated in the statespace. The goal, again, is to maximize the expected reward over the T steps as in the previous problem.

Delayed feedback was introduced by Anderson [4] in an early work in mid 1960s. Since then, though there have been additional results [32,10], a theoretical guarantee on adaptive decision making under delayed observations has been elusive, and the computational difficulty in obtaining such has been commented upon in [12,5,29,13]. One recent example of this problem arose in information gathering and polling [1]. The notion of delays are also discussed in context of adaptive query processing [33], reflective control in cloud services [11], or unmanned aerial vehicles [28]. Observe that in the presence of budgets we cannot play an arm continuously without considering the obtained reward or using clever estimation techniques.

The two problems are similar, even though they appear to be very different. In the first problem the reward derived from an arm is not just the function of the current play of the arm, because the other arms may have larger or smaller values. In the latter, the information derived about an arm is not a function of

[2] Since the underlying true object is a distribution, the specific sequence of observation is not important.

[3] The authors of [31] consider a similarly named problem but that paper maximizes the maximum value seen at any time across any arm and does not consider budgets.

the current play of the arm — we may have delayed information about arm i due to a previous play δ_i steps ago even when we are playing arm j currently. It may therefore appear that arm i is more interesting than arm j' – even though we did not play either i, j'. Observe that this entanglement of decisions is sufficient to make the MaxMAB problem Max-SNP hard even if the distributions are known, that is the priors are completely resolved (stochastic optimization case, [18]). To achieve a $O(1)$ approximation for even this special case one needs to solve submodular optimization under knapsack constraints [27]. However if the distributions are unknown, neither of Problem 1 or 2 is submodular – both of them violate the **idling bandit property**; that is, the reward obtained from an arm in any state is not affected by *when* the arm is scheduled for play. The idling bandit property is at the core of all existing analysis of stochastic (including both Bayesian and prior free problems) MAB problems with provable guarantees [26,6,17,8]. The exchange properties required for defining submodularity as well as its extensions such as sequence or adaptive submodularity [30,19,3] imply the idling bandit property. It may appear that the issue can be avoided by appealing to non-stochastic/adversarial MABs [7] – but it is well known that online convex analysis cannot be analyzed well in the presence of large number of arms and a "state" that couples the time steps [14]. Budgets, delays and concave reward functions all enforce such a state. In light of the above, the goal of the paper is to design algorithms that produce provably near optimal policies for the two problems in polynomial time. We show:

Theorem 1. *The* MaxMAB *problem has a $O(1)$-approximate (almost) index policy which can be found in polynomial time (polynomial in $\sum_i |\mathcal{S}_i|, n$ and T).*

Theorem 2. *Assuming $\delta_i = o(T/\log T)$, in polynomial time we can find an $O(1)$-approximate (almost) index policy for delayed feedback.*

The standard of efficiency in the MAB context is an algorithm which is polynomial in $\sum_i |\mathcal{S}_i|, n$ and T. Standard computation of index policies in literature use dynamic programming over single arm state spaces $\mathcal{S}_i$ for each i, and have running time which is polynomial in $\sum_i |\mathcal{S}_i|, n$ and T, see [17,8]. If the number of per-arm outcomes is a small constant, (e.g., success or failure of treatment, click or no click, delivery or failure of packets) each $\mathcal{S}_i$ is polynomial in T.

Technical Contributions: We follow a template that has been used in [21]; where the optimization is expressed as a relaxed decision problem where the goal is to find a collection of *single-arm* policies (whose states and actions are defined only for a single arm). Note that the solution to the relaxation will not be feasible since it need not encode many constraints that make a policy feasible – for instance, it will not encode that only T plays are allowed in any realization of the reward distributions. The goal will be to combine these single arm policies in a meaningful manner. This suggests a recipe which can be stated as follows: (i) Identify a tractable state space for single arms, which is a function of $\mathcal{S}_i$, (ii) Reason that good single arm policies exist in that space, (iii) Find such policies feasible for a small set of *coupling constraints* (using step (i)) and (iv) Schedule the single arm policies into a globally feasible policy.

Step (iii) will often involve the use of convex programming or Lagrangians – these are standard. The difficulty is that step (iii) has to be performed on a suitable state space as identified in (i), proved effective in (ii), and proved easy to schedule as in (iv). The technical complexity of the paper lies in steps (i), (ii), and (iv), which we discuss below:

1. **Succinct Expression of Policies:** For the problems considered above, the state of the policy at any time step is complicated, and does not lend to an easy way of performing steps (i) and (ii). For instance, in the MaxMAB problem, the reward depends on the specific set of K arms that are played at any step. This does not correspond naturally to any policy whose execution is restricted to a *single arm*. Similarly, in the delayed feedback problem, the state for a *single arm* depends on the time steps in the past where plays were made and the policy is awaiting feedback – this is exponential in the length of the delay. Our first contribution is to show that in each case, there are different state space for the policy which has size polynomial in $\sum_i |\mathcal{S}_i|, T$, over which we can write a relaxed decision problem.
2. **Index policies and Scheduling:** A second technically novel step is the combination of single-arm policies into a globally feasible policy. The single arm policies derived from the previous step do not preserve their accrued reward when they are combined in a naive fashion. We show techniques for *scheduling* these policies so that we can still obtain a good fraction of the reward. This aspect is surprising for both the problems we consider: For MaxMAB, the relaxation does not even encode that we receive the max reward every step, and instead only captures the *sum of rewards* over time steps. Yet, we can schedule the single-arm policies so that the max reward at any step yields a good approximation. For delayed feedback, the single-arm policies wait different amounts of time before playing, and we show how to interleave these plays to obtain good reward. We note that scheduling ideas have been independently applied to MABs in [24]; but that setting is different and we need newer ideas for concave rewards.

One final technical issue is that the process of scheduling single-arm policies might violate the feasibility of the time horizon. We crucially need a Truncation Theorem which we present next. A very similar claim is provided by Farias and Madan[15, Lemma 2]; where the proof is provided by induction. We provide a slightly modified statement since we use the theorem for delayed feedback settings as well which creates some (but minor) complications for the inductive argument. The details can be found in the full version [22].

Theorem 3. *(Truncation Theorem) Given any arbitrary single arm policy $\mathcal{P}$ (see Definition 4) which traces out a path over the state space $\mathcal{S}$ (does not gain information magically) and where the outcomes at each step are drawn from an underlying non-negative reward distribution, consider a policy $\mathcal{P}'$ that is identical to $\mathcal{P}$ on each decision path but stops early, such that $\mathcal{P}'$ makes at least β fraction of the plays made by $\mathcal{P}$ on any decision path. Then (i) $R(\mathcal{P}') \geq \beta R(\mathcal{P})$ and (ii) $\mathcal{T}(\mathcal{P}') \leq \mathcal{T}(\mathcal{P})$ where $R(\mathcal{P}), \mathcal{T}(\mathcal{P})$ are the reward and plays (both in expectation).*

2 Preliminaries

2.1 Priors and State Spaces

We have already discussed the Bayesian MAB setting in Definition 1. Playing the arm in state u yields a transition to state v with probability $\mathbf{p}_{uv}$, provided v can be obtained from u in one additional observation; the probability $\mathbf{p}_{uv}$ is simply the probability of this observation conditioned on the posterior X_{iu} at u. The expected posterior mean at a state $u \in \mathcal{S}_i$, denoted by $r_u = \mathbf{E}[X_{iu}]$ satisfies a Martingale property $r_u = \sum_v \mathbf{p}_{uv} r_v$.

Definition 2. *Given a random variable X define* $\textsc{Tail}(X, \lambda) = \sum_{x \geq \lambda} \textsc{Pr}[X = x] \cdot x$. *Define* $\textsc{Excess}(X, \lambda) = \sum_{x > \lambda} \textsc{Pr}[X = x] \cdot (x - \lambda)$.

It is easy to see that both $\textsc{Tail}(X_{iu}, \lambda), \textsc{Excess}(X_{iu}, \lambda)$ are martingales over $\mathcal{S}_i$.

Example Priors and State Spaces. Consider an example $\mathcal{S}_i$ that arises from a fixed but unknown distribution over two outcomes (success or failure of treatment, click or no click, conversion or no conversion, delivery or failure). This is a Bernoulli$(1, \theta)$ trial where θ is unknown. The states are represented with parameters $\alpha_0, \alpha_1 \in \mathbb{Z}^+$; and correspond to the distribution $Beta(\alpha_1, \alpha_0)$ whose p.d.f. is of the form $c\theta^{\alpha_1 - 1}(1-\theta)^{\alpha_0 - 1}$, where c is a normalizing constant. Given the distribution $Beta(\alpha_1, \alpha_0)$ as our prior, the expected value of θ is $\frac{\alpha_1}{\alpha_1 + \alpha_0}$. On seeing a 1, the posterior (of the current sample, and the prior for the next sample) is updated to $Beta(\alpha_1 + 1, \alpha_0)$. On seeing a 0, the new distribution is $Beta(\alpha_1, \alpha_0 + 1)$. Note the hard case for any algorithm (and the case that does arise in practice) is when the input to the problem is a set of arms $\{i\}$ with priors $\mathcal{D}_i \sim Beta(\alpha_{1i}, \alpha_{0i})$ where $\alpha_{0i} \gg \alpha_{1i}$ which corresponds to a set of poor prior expectations of the arms. For a horizon of size T, the entire information is captured by a DAG of size $O(T^2)$. The example naturally extends to Dirichlet priors which are conjugate priors of multinomials and generalize Beta distributions for multiple outcomes.

2.2 Decision Policies and Single-arm Policies

Definition 3. *A decision* **policy** *is a mapping from the current state of the overall system to an* action*, which involves playing K arms.*

We use the term *state* both to denote $\mathcal{S}_i$, which is the state corresponding to the posterior of the arm, as well as the state of the decision policy, which could involve the joint states $\{u \in \mathcal{S}_i\}$ of all the arms at any step, the remaining time horizon, remaining budgets, and other variables (such as outstanding feedback of all arms in the delayed feedback version). Therefore, the "state" used by a decision policy could be much more complicated than the product space of $\mathcal{S}_i$.

Definition 4. *Given an execution of the global policy $\mathcal{P}$, define its* **projection** *on arm i to be the policy P_i defined by the actions induced on $\mathcal{S}_i$; we term this a*

single-arm *policy. Note that the global policy may take an action in arm i based on information regarding other arms (an entangled state) – that side information is lost in the projection.*

In a single-arm policy for the MaxMab problem, we can further compress idle-time of an arm as follows: If the global policy plays arm i, and then waits t steps before playing next (because it was playing other arms), in the projection, we make these plays consecutive. By the bandit property, the outcome of the next play is stochastically preserved in this process. Therefore, the state of the policy P_i is only captured by the state $u \in \mathcal{S}_i$ and the outstanding budget. This is not feasible in the delayed feedback setting.

Definition 5. *A policy $\mathcal{P}$ is a decomposable* **index policy**, *if each arm is reduced to a single* priority *value based on current state of that arm and possibly a few global parameters. The feasible set with the highest priority is played in every step.*

2.3 Modeling Budgets

One ingredient in our problem formulations is natural budget constraints in individual arms. As we will see later, these constraints introduce non-trivial complications in designing the decision policies. There are three types of budget constraints we can consider.

Play Budget: This corresponds to the number of times an individual arm i can be played, typically denoted by T_i where $T_i < T$. In networking, each play attempts to add traffic to a route and we may choose to avoid congestion. In online advertising, we could limit the number of impressions of a particular advertisement, often done to limit advertisement-blindness.

Observed Value Budget: The total obtained from arm i should be at most its budget B_i.

Achieved Reward Budget: In MaxMab, the reward of only the arm that is maximum is used, and hence only this budget is depleted. In other words, there is a bound on the total reward achievable from an arm (call this A_i), but we only count reward from an arm when it was the maximum value arm at any step.

Observed that the play and observed value budgets simply involve truncating the state space $\mathcal{S}_i$, so that an arm cannot be played if the number of plays or observations violates the budget constraint. The achieved reward budget is more complicated to handle, and perhaps the most relevant since arms that "lost out" (were not the desired ones) should not be charged. We discuss only the achieved reward budget in Section 3. However the observed value budget, even for $K = 1$ simultaneous plays is nontrivial in the context of delays in feedback, we discuss that model in Section 4 further.

3 Problem 1: MaxMAB and Concave Objectives

In MaxMAB, at each step, the decision policy can play at most K arms but the reward obtained is the maximum of the values that are observed. In other words, the policy plays K arms each step, but is allowed to choose the arm with the maximum observed value and obtain its reward. Note that the states of all K arms evolve to their respective posteriors, since all arms were observed. We assume that only the chosen arm loses its budget – otherwise the budgets can be incorporated in the state space. To simplify the discussion, we make two assumptions, but these assumptions do not lose any generality. First we assume that all reward values q are powers of two. This assumption loses a factor of 2 in the approximation ratio, since we can round q down to powers of 2 while symbolically maintaining their distinction in the prior updates. Second we assume for any q observed by arm i we have $q \leq B_i$. We allow the reward of the last step to count completely – the final policy stops when the budget is B_i, thereby losing at most a factor 2 in reward.

Extending the choice to single arm policies: Recall that the goal would be to devise global policies that arise from combination of single arm policies. In this setting, the state of a single-arm policy is captured by the state $u \in \mathcal{S}_i$ as well as the outstanding budget. Note that a global policy can play an arm, but only obtains reward if the arm is *chosen* as the maximum. Since we project onto the behavior of a single arm, we distinguish between a "play" of an arm, and a "choice", which is the subset of plays that yield reward. Therefore, a single-arm policy for arm i takes one of several actions in each state: (i) Stop execution; (ii) Play arm i; (iii) Choose arm i after the arm has been played, yielding reward. If the arm is chosen when the observed value from that play is q then we collect a reward q. The state evolves whenever the arm is played.

Definition 6. *The single-arm policy $\mathcal{P}_i$ is considered* feasible *if the total reward on any decision path (taking into account the "choice" action) is at most B_i. Let the set of feasible policies for arm i be $\mathcal{F}_i$.*

A modified goal is therefore to find a collection of *feasible* single-arm policies $\mathcal{P}_i$.

The Overall Optimization Approach: We will use the four fold recipe suggested in the introduction. We express the problem as an optimization problem over policies and then show that *after a few steps of relaxations* we can express the resultant as natural linear optimization problems; which can then be solved using many different techniques including techniques including existing index policies. The steps of relaxation indicate which states in the single arm state space are relevant, albeit with a loss of approximation factor.

Definition 7. *Let $\mathcal{T}(\mathcal{P}_i)$ be the expected number of plays of the policy $\mathcal{P}_i$.*

Definition 8. *For any single-arm policy $\mathcal{P}_i$, let $N(q, \mathcal{P}_i)$ denote the expected number of times the policy observes the value q and chooses arm i.*

$$OPT \leq \text{LPMaxMab} = \max_{\{\mathcal{P}_i \in \mathcal{F}_i\}} \sum_i \sum_q qN(q, \mathcal{P}_i)$$
$$s.t. \quad \sum_i \sum_q N(q, \mathcal{P}_i) \leq T \text{ and } \sum_i \mathcal{T}(\mathcal{P}_i) \leq KT$$

The Lagrangian of *only the first constraint* gives us:

$$\text{LagMaxMab}(\lambda) = \max_{\{\mathcal{P}_i \in \mathcal{F}_i\}} \lambda T + \sum_i \sum_q (q - \lambda) N(q, \mathcal{P}_i) \; s.t. \; \sum_i \mathcal{T}(\mathcal{P}_i) \leq KT$$

Steps 1 and 2: Suitable subset of states: In the absence of any constraint the optimal policy for every arm i, has the following property: If it plays the arm in state $u \in \mathcal{S}_i$, then it chooses the arm if the observed reward is $q \geq \lambda$. In presence of budgets, such a fact is not true — observe that the $N(q, \mathcal{P}_i)$ is determined *after* $\mathcal{P}_i$ is chosen. *Not choosing to take a reward immediately allows us to explore a larger state space.* The next definition and lemma provide a characterization of the states and observations that are useful.

Definition 9. *For any parameter ν, define the truncated state space $\mathcal{S}_i(\nu)$ as follows. The state u is retained in the truncated state space iff the total value from all observations of $q \geq \nu$ is at most B_i — the horizon depends on ν (but still at most T).*

Lemma 1. *For any $\lambda \geq 0$, there exists $\{\mathcal{P}_i\}$ such that (i) The policy $\mathcal{P}_i$ runs on state space $\mathcal{S}_i(\nu_i(\lambda))$, and (ii) It always chooses the arm if the reward is $\geq \nu$ and (iii) $\{\mathcal{P}_i\}$ provide a 4 approximation to* LagMaxMab(λ).

Proof. Define q to be small if $q \in [\lambda, 2\lambda)$ and large otherwise. Since q values are powers of 2, there is only one small value of q; call this value q_s. Consider the optimal solution to LagMaxMab(λ). Either half the objective value in LagMaxMab(λ) is achieved by choosing q_s (**Case 1**) or by choosing large $q > q_s$ (**Case 2**). The policies will be different depending on the two cases — the algorithm need not know which case applies; we will solve both cases and choose the better solution. In **Case 1**, we set $\nu = \lambda$, and in **Case 2** we set $\nu = \min\{q | q \geq 2\lambda\}$. Consider the collection of optimal policies for LagMaxMab(λ). Modify each policy $\mathcal{P}_i$ as follows: Keep playing as long as the state is feasible in $\mathcal{S}_i(\nu)$; otherwise stop playing. Clearly the new policy is also feasible for the constraint $\sum_i T(\mathcal{P}_i) \leq KT$ since the original policy was. In the new policy, whenever a reward observed is at least ν choose the arm even if the original policy had not chosen it. We now bound the value of the new policy by considering every decision path of the original policy and considering the contribution to the objective of LagMaxMab(λ) in this decision path.

Case 1: Suppose choosing q_s contributes more than half of the optimal objective. In this case, $\nu = \lambda$. Consider just the contribution of $q_s - \lambda$ on any decision path, and suppose q_s is chosen k times, so that $kq_s \leq B_i$. If the modified policy chooses this q for $k' \geq k$ times, then its value clearly dominates the original

policy's contribution. Otherwise, suppose the modified policy chooses q_s some $k' < k$ times, then the choices of the remaining q must have exhausted the budget. Since per unit q, the policy generates value at least $1 - \lambda/q_s$, a knapsack argument shows that the value generated in the modified policy on all q is at least the value generated by the original policy on the $q_s - \lambda$ values which is at least:

$$B_i \left(1 - \frac{\lambda}{q_s}\right) \geq k q_s \left(1 - \frac{\lambda}{q_s}\right) = k(q_s - \lambda)$$

Note that the contribution from q_s in the original policy is $k(q_s - \lambda)$.

Case 2: Suppose the large q values contribute to at least half the optimal objective. In this case, we have $\nu = \min\{q | q \geq 2\lambda\}$. On any decision path, consider just the contribution of large $q - \lambda$. Note that $q - \lambda \geq q/2$ for such q, so that in the worst case, the new policy exhausts its budget and contributes $B_i/2$ to the objective while the original policy could have contributed B_i. If the new policy does not exhaust its budget, it must match the value of the original policy on large q. We lose a factor two in splitting the analysis into two cases, and a factor of 2 within the second case. Therefore, the new policy is a 4 approximation and satisfies the properties in the statement. □

Step 3: Finding the single arm policies. The next theorem provides us the policies based on approximately optimal Lagrangian solutions, see [25]. We need to use that idea twice.

Theorem 4. *In time polynomial in $\sum_i \mathcal{S}_i$, T we can find λ^* and policies $\{\mathcal{P}_i(\lambda^*)\}$ with associated $\{\nu_i(\lambda^*)\}$ that satisfy the properties in Lemma 1 and that are a $16(1+\epsilon)$-approximation to the solution to* LPMaxMab.

Proof. We will use Lagrangians twice. First, fix λ. Observe that under the characterization of Lemma 1, we always choose any value which is $\nu(\lambda)$ or above. So consider

$$\textsc{LagMaxMab}(\lambda, \zeta) = \max_{\{\mathcal{P}_i \in \mathcal{F}_i\}} \lambda T + \zeta K T + \sum_i \left(\sum_q (q - \lambda) N(q, \mathcal{P}_i) - \zeta \mathcal{T}(\mathcal{P}_i) \right)$$

Thus $\textsc{LagMaxMab}(\lambda, \zeta)$ is completely an unconstrained policy; and the solution is easy if we know $\nu(\lambda)$. We can compute the solution by a bottom up dynamic programming (DP). At each node u we have the policies for all its children available (using the subpolicies constructed by the DP so far). The reward available at node u is $\sum_{q \geq \nu(\lambda)} (q - \lambda) Pr[X_{iu} = q]$ where X_{iu} is the value distribution at node u. The cost of playing at node u is ζ. For each child node v which we reach with probability p_{uv} suppose the subpolicy is P_v and its reward (from values $\nu(\lambda)$ or above) is $R(P_v)$. Then the decision at u is if:

$$\sum_v p_{uv} \left(R(P_v) - \zeta \mathcal{T}(P_v)\right) + \sum_{q \geq \nu(\lambda)} (q - \lambda) Pr[X_{iu} = q] > \zeta$$

If the answer is yes, then we will play at node u and otherwise we will not play. Once we compute the decision at the root of $\mathcal{S}_i$ we are done.

However we have a slight problem – we know $\nu(\lambda)$ is one of two values! We repeat the computation for both and choose the better policy for each arm. Let this policy be P_i. Therefore the objective function:

$$\textsc{LagMaxMab}(\lambda, \zeta) \geq \frac{1}{4}\textsc{LagMaxMab}(\lambda) + \zeta \left(KT - \sum_i \mathcal{T}(P_i) \right)$$

We now observe that for $\zeta = 0$ if we have $\sum_i \mathcal{T}(P_i) \leq KT$ we are done. Let us assume $\sum_i \mathcal{T}(P_i) > KT$. For $\zeta = \infty$ we will never make a play and $\sum_i \mathcal{T}(P_i) = 0$. We can now binary search over ζ such that we have two values $\zeta^+ > \zeta^-$ and for ζ^+ we have the policies $\{P_i^+\}$ satisfying $\sum_i \mathcal{T}(P_i^+) < KT$ and for ζ^- we have the policies $\{P_i^-\}$ satisfying $\sum_i \mathcal{T}(P_i^-) > KT$. Note at equality, we can simply return the policies; and the $\zeta\left(KT - \sum_i \mathcal{T}(P_i)\right)$ term will vanish. We perform the binary search till $|\zeta^+ - \zeta^-|$ is at most $\frac{\epsilon}{4nKT}$ times the maximum reward of any arm (which is a lower bound on $\textsc{LagMaxMab}(\lambda)$). Now if we take a convex combination $a_+ \sum_i \mathcal{T}(P_i^+) + a_- \sum_i \mathcal{T}(P_i^-) = KT$ then the reward of the convex combination of the two policies will be a $4+\epsilon/2$ approximation (for small $\epsilon \ll 1$). Observe that either half of the contribution arises from the $\{P_i^+\}$ being chosen with probability a^+ or half of the contribution arises from the $\{P_i^-\}$ being chosen with probability a^-. This gives a randomized $8 + \epsilon$ approximation to $\textsc{LagMaxMab}(\lambda)$.

We now apply the same idea to λ and $\sum_i \sum_q N(q, \mathcal{P}_i)$. If for $\lambda = 0$ we have $\sum_i \sum_q N(q, \mathcal{P}_i) \geq T$ then we already have a feasible solution and therefore we can assume that $\sum_i \sum_q N(q, \mathcal{P}_i) > T$ for $\lambda = 0$. Now is λ is the largest possible value we immediately know that $\sum_i \sum_q N(q, \mathcal{P}_i) = 0$ since there will no benefit to choosing any value. We perform a binary search till $|\lambda^+ - \lambda^-| \leq \frac{\epsilon}{T}$ times the maximum reward from any single arm (which is at most $\frac{\epsilon}{nT}$ times $\text{LP}\textsc{MaxMab}$). Once again we lose another factor 2 and the theorem follows. □

Step 4: Scheduling and Designing the Final (almost) Index Policy: Consider the single-arm policies $\mathcal{Q}_i$ constructed in Theorem 4. Modify these policies so that they stop executing after $T/2$ steps. Choose a subset S_0 of arms, where each arm is placed in S_0 independently with probability $\frac{1}{24}$. Order the arms in S_0 arbitrarily. Each arm in S_0 is in one of three possible states: Ready, Current, and Finished. Initially, all arms are Ready. Make the first K arms Current, and denote the set of Current arms as S. We execute the policies $\mathcal{Q}_i$ for $i \in S$ as described below. Whenever a policy terminates, we mark this arm as Finished, remove it from S, and place any Ready arm in S. At any step, suppose the arms in S are $i_1, \ldots, i_K$ in states $u_1, \ldots, u_K$. Then the policy is below (observe that one of the three is always true); for reward always choose the arm with the maximum observed value.

All: If $\sum_{s=1}^{K} \Pr[X_{i_s u_s} \geq \nu_{i_s}(\lambda^*)] \leq \frac{2}{3}$ then we play all the arms.
Stall: Play i_s such that $\Pr[X_{i_s u_s} \geq \nu_{i_s}(\lambda^*)] \geq \frac{1}{3}$.
Throttle: Play a subset S' such that $\sum_{s \in S'} \Pr[X_{i_s u_s} \geq \nu_{i_s}(\lambda^*)] \in \left[\frac{1}{3}, \frac{2}{3}\right]$.

Lemma 2. *If the policies $\mathcal{Q}_i$ are modified to stop execution after $T/2$ steps, this reduces their expected reward by at most a factor of 2.*

Proof. Given a random variable X let $\textsc{Tail}(X,\nu) = \sum_{x \geq \nu} \Pr[X = x] \cdot x$. Since $\textsc{Tail}(X_{iu}, \nu)$ is a martingale over $\mathcal{S}_i(\nu)$, the lemma now follows by applying Theorem 3 to $\mathcal{S}_i(\nu_i(\lambda^*))$. □

Lemma 3. *Conditioned on $j \in S_0$, the policy $\mathcal{Q}_j$ executes to completion with probability at least $\frac{1}{2}$ on* all *its paths.*

Proof. To bound the probability of this event, assume arm j is placed last in the ordering, and only marked Current *after* all other policies are marked Finished - this only reduces the probability that $\mathcal{Q}_j$ executes to completion on any of its decision paths. For the arms $i \neq j$, observe that at each step, either at least K arms are played, or sufficiently many arms are played so that the sum of the probabilities that the observed value of arm i exceed $\nu_i(\lambda^*)$ is at least $\frac{1}{3}$. Therefore, at each step, if S is the set of arms marked Current, Z_{it} is an indicator variable denoting whether arm i is played, and W_{iqt} is an indicator variable denoting whether arm i is observed in state q, we must have:

$$\sum_{i \in S} \left(\sum_{q \geq \nu_i(\lambda^*)} W_{iqt} + \frac{Z_{it}}{K} \right) \geq \frac{1}{3}$$

If Y denotes the random variable corresponding to the total time for which policies for arms $i \neq j$ execute, by linearity of expectation:

$$\begin{aligned} \frac{\mathbf{E}[Y]}{3} &\leq \mathbf{E}\left[\sum_t \sum_{i \neq j} \left(\sum_{q \geq \nu_i(\lambda^*)} W_{iqt} + \frac{Z_{it}}{K} \right) \right] \\ &= \mathbf{E}_{S_0}\left[\sum_{i \in S_0, i \neq j} \left(\sum_q N(q, \mathcal{Q}_i) + \frac{\mathcal{T}(\mathcal{Q}_i)}{K} \right) \right] \leq \frac{2T}{24} \end{aligned}$$

Therefore, $\mathbf{E}[Y] \leq T/4$ so that $\Pr[Y \leq T/2] \geq 1/2$ by Markov's inequality. In this event, $\mathcal{Q}_j$ executes to completion, since its horizon is at most $T/2$. □

Lemma 4. *Consider the event $j \in S_0$ and j is marked Finished. In this event, suppose we count the contribution from this arm to the overall objective only when it is the only arm that observes a value larger than $\nu_j(\lambda^*)$. Then, the expected contribution from arm j is at least $\frac{1}{3}$ times the value of policy $\mathcal{Q}_j$.*

Proof. We have $\sum_i \Pr[X_{iu} \geq \nu_i(\lambda^*)] \leq \frac{2}{3}$ whenever the arm j is played simultaneously with any other arm. Since the other arms are independent of arm j, if j is observed at $q \geq \nu_j(\lambda^*)$ then with probability at least $\frac{1}{3}$ all other arms i are observed to be less than the respective $\nu_i(\lambda^*)$. □

Combining Theorem 4, and Lemmas 2, 3, and 4, we observe that with constant probability $\mathcal{Q}_j$ executes to completion on all its decision paths, and in this event, has expected value a constant factor of the LP contribution. By linearity of expectation, we have the following theorem;

Theorem 1. *The* MaxMAB *problem has a $O(1)$-approximate policy which is a throttled index policy. The policy can be found in time polynomial in $\sum_i |\mathcal{S}_i|, T$.*

4 Problem 2: Delayed Feedback

In this problem if an arm is played, the feedback about the reward outcome is available only after δ_i time steps. For simplicity of exposition assume that we are playing one arm at a time, i.e., $K = 1$. As a consequence we are in a simpler model regarding budgets – the only constraint we now have is that the total reward from arm i is at most B_i. Define $t_i = T/\delta_i$. We assume that $\delta_i = o(T/\log T)$, which implies that the horizon is slightly separated from the delays and $t_i = \omega(\log \delta_i)$. We show the following theorem. In the interest of space, the details of the proof are in the full paper [22].

Theorem 2. *Assuming $\delta_i = o(T/\log T)$, there is a constant factor approximation to the Finite Horizon MAB problem with delayed feedback. The running time for computing this policy is poly$(T, \sum_i |\mathcal{S}_i|)$.*

We show the above theorem by identifying a tractable state space for single arms, which is a function of $\mathcal{S}_i$; reasoning that good single arm policies exist in that space and computing them using linear programming; and finally scheduling the single arm policies into a globally feasible policy. We use LPDelay to bound of the reward of the best collection of single-arm policies - the goal is to find one policy per arm so that the total expected number of plays is at most T and the expected reward is maximized.

Definition 10. *Let $\mathcal{C}_i(T)$ be the set of all single-arm policies for arm i over a horizon of T steps, obeying the above budget constraint.*

Lemma 5. $OPT \leq \text{LPDelay} = \max_{\{P_i \in \mathcal{C}_i(T)\}} \{\sum_i R(P_i) \mid \sum_i \mathcal{T}(P_i) \leq T\}$

However, in the case of delayed feedback, describing a single-arm policy is more complicated. This policy is now a (randomized) mapping from the current state of the arm to one of the following actions: (i) make a play; (ii) wait some number of steps (less or equal to T), so that when the result of a previous play is known, the policy changes state; (iii) wait a few steps and make a play (without extra information); or (iv) quit. The *state* of the system is now captured by not only the current posterior $u \in \mathcal{S}_i$, but also the plays with outstanding feedback and the remaining time horizon. Note that the state encodes plays with outstanding feedback, and this has size 2^{δ_i}. Therefore, it is not even clear how to even express LPDelay in polynomial space. The main idea is to perform a sequence of transformations that will show that for any single-arm policy in $P_i \in \mathcal{C}_i(T)$, there is a different single-arm policy that is much more structured than P_i that achieves at least a constant-factor of the reward of P_i.

Denote single-arm policies for arm i restricted to a horizon T as $\mathcal{P}(i, T)$.

Step 1: Block Structured Policies. We first show that all single-arm policies can be replaced with block structured policies of the following form:

Definition 11. *A single-arm policy is said to be* Block Structured *if the policy executes in phases of size* $(2\delta_i + 1)$. *At the start of each phase (or block), the policy makes at most* $\delta_i + 1$ *consecutive plays. The policy then waits for the rest of the block in order to obtain feedback on these plays, and then moves to the next block. A block is defined to be* full *if exactly* $\delta_i + 1$ *plays are made in it.*

Lemma 6. *Any policy* $\mathcal{P}(i, T)$ *can be converted it to a Block Structured policy* $\mathcal{P}'(i, 2T)$ *such that* $R(\mathcal{P}(i, T)) \leq R(\mathcal{P}'(i, 2T))$ *and* $\mathcal{T}(\mathcal{P}'(i, 2T)) \leq \mathcal{T}(\mathcal{P}(i, T))$

The idea behind this proof is simple – we simply insert delays of length δ_i after every chunk of plays of length δ_i.

Step 2: Well-Structured Policies. We now define a compact *delay free* policy:

Definition 12. *Define a block-structured policy to be* c-delay-free *for* $c \leq 1$ *if the first time the policy encounters a block with at least* $c\delta_i$ *plays, it plays every step (without waiting) beyond this point (using feedback from* δ_i *plays ago) until it stops executing.*

Lemma 7. *Given any Block Structured policy* $\mathcal{P}(i, 2T)$ *we can construct a* c*-delay-free Block-structured policy* $\mathcal{P}'(i, 2T)$, *such that* $R(\mathcal{P}(i, 2T)) \leq R(\mathcal{P}'(i, 2T))$ *and* $\mathcal{T}(\mathcal{P}'(i, 2T)) \leq (1 + \frac{1}{c})\mathcal{T}(\mathcal{P}(i, 2T))$.

The proof uses the idea of *simulation* - we make more plays initially, but hold on to the outcomes of the extra plays. When the original policy makes plays in a subsequent block, we eliminate these plays and instead use the outcomes of the saved up plays. We next use the proof idea of Lemma 7 recursively to prove Lemma 8. Note, any policy uses at most $\frac{T}{\delta_i} = \omega(\log \delta_i)$ blocks.

Definition 13. *For constant* $\alpha < 1$, *define a* c*-delay-free block-structured single-arm policy* $\mathcal{P}$ *to be* (α, c)-well-structured *if after encountering at most* $q = (\alpha + o(1))t_i$ *blocks, the policy switches to playing continuously. Here* $t_i = T/\delta_i$.

Lemma 8. *For any* $\alpha < 1$ *and* $c \leq \frac{\alpha}{\alpha+2}$, *given a* c*-delay-free policy* $\mathcal{P}(i, 2T)$, *there is a* (α, c)*-well-structured policy* $\mathcal{P}'$ *such that* $R(\mathcal{P}(i, 2T)) \leq R(\mathcal{P}'(i, 2T))$ *and* $\mathcal{T}(\mathcal{P}'(i, 2T)) \leq (1 + \frac{2}{\alpha})\mathcal{T}(\mathcal{P}(i, 2T))$.

Step 3: Finding Well-Structured Policies: We prove the following theorem.

Theorem 5. LPDelay *has a* $1/\alpha$*-approximation over* $(\alpha, \frac{\alpha}{\alpha+2})$*-well structured policies for* $\alpha \leq 1/8$ *truncated to a horizon of* $T/2$. *The new LP has the relaxed constraint:* $\sum_i \mathcal{T}(\mathcal{P}_s(i, T/2)) \leq \gamma T$ *where* $\gamma = 2(1 + 1/\alpha)(1 + 2/\alpha)$.

The LP yields one randomized well-structured policy $\mathcal{P}^r(i, T/2)$ for each arm i. This policy is described as: If the state at the beginning of a block is σ then

1. Choose ℓ with probability $p(i, \sigma, \ell)$, and make ℓ plays in the current block.
2. Wait till the end of the block; obtain feedback for the ℓ plays; update state.

Step 4: Priority Based Combining. At this point we have a collection of randomized policies $\mathcal{P}^r(i, T/2)$ s.t. $\sum_i \mathcal{T}(\mathcal{P}^r(i, T/2)) \leq T$ and $\sum_i R(\mathcal{P}^r(i, T/2)) = \Omega(OPT)$. We now show how to combine these into a globally approximate and feasible solution. Since multiple policies must remain active in the combination, we need a novel priority based scheme for combining the policies.

Consider the execution of $\mathcal{P}^r(i, T/2)$. We describe the arm as *active* if it is either making plays or is at the beginning of a block where it can make plays; and *passive* if it is waiting for feedback on the plays within the block. Any arm which completed its waiting for the feedback turns from passive to active mode. The final policy is:

1. Choose an arbitrary order the arms $\{i\}$ denoted by π. Each arm "participates" with probability 1/4. Initially, all participating arms are active.
2. On each new play, among all participating arms:
   (a) Find the lowest rank arm, say i', that is active.
   (b) Allocate the current play to i' according to the policy $\mathcal{P}^r(i', T/2)$. (As a result the arm may become passive and wait for feedback.)

Lemma 9. *The expected contribution of i from the combined policy is at least $R(\mathcal{P}^r(i, T/2))/8$.*

The above lemma along with Theorem 5, prove Theorem 2 by linearity of expectation, concluding the analysis.

Acknowledgements. We thank Martin Pál for many useful discussions.

References

1. Agarwal, D., Chen, B.-C., Elango, P.: Explore/exploit schemes for web content optimization. In: Proceedings of ICDM, pp. 1–10 (2009)
2. Akella, A., Maggs, B., Seshan, S., Shaikh, A., Sitaraman, R.: A measurement-based analysis of multihoming. In: Proceedings of SIGCOMM, pp. 353–364 (2003)
3. Alaei, S., Malekian, A.: Maximizing sequence-submodular functions and its application to online advertising. CoRR abs/1009.4153 (2010)
4. Anderson, T.W.: Sequential analysis with delayed observations. Journal of the American Statistical Association 59(308), 1006–1015 (1964)
5. Armitage, P.: The search for optimality in clinical trials. International Statistical Review 53(1), 15–24 (1985)
6. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning 47(2-3), 235–256 (2002)
7. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.: The nonstochastic multiarmed bandit problem. SIAM J. Comput. 32(1), 48–77 (2003)
8. Bertsekas, D.: Dynamic Programming and Optimal Control, 2nd edn. Athena Scientific (2001)
9. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning and Games. Cambridge University Press (2006)
10. Choi, S.C., Clark, V.A.: Sequential decision for a binomial parameter with delayed observations. Biometrics 26(3), 411–420 (1970)

11. Demberel, A., Chase, J., Babu, S.: Reflective control for an elastic cloud appliation: An automated experiment workbench. In: Proc. of HotCloud (2009)
12. Ehrenfeld, S.: On a scheduling problem in sequential analysis. The Annals of Mathematical Statistics 41(4), 1206–1216 (1970)
13. Eick, S.G.: The two-armed bandit with delayed responses. The Annals of Statistics 16(1), 254–264 (1988)
14. Even-Dar, E., Kearns, M., Wortman, J.: Risk-sensitive online learning. In: Balcázar, J.L., Long, P.M., Stephan, F. (eds.) ALT 2006. LNCS (LNAI), vol. 4264, pp. 199–213. Springer, Heidelberg (2006)
15. Farias, V.F., Madan, R.: The irrevocable multiarmed bandit problem. Operations Research 59(2), 383–399 (2011)
16. Gittins, J., Glazebrook, K., Weber, R.: Multi-Armed Bandit Allocation Indices. Wiley (2011)
17. Gittins, J.C., Jones, D.M.: A dynamic allocation index for the sequential design of experiments. In: Progress in Statistics (European Meeting of Statisticians) (1972)
18. Goel, A., Guha, S., Munagala, K.: How to probe for an extreme value. ACM Transactions of Algorithms 7(1), 12:1–12:20 (2010)
19. Golovin, D., Krause, A.: Adaptive submodular optimization under matroid constraints. CoRR abs/1101.4450 (2011)
20. Guha, S., Munagala, K.: Approximation algorithms for budgeted learning problems. In: Proc. of STOC (2007)
21. Guha, S., Munagala, K.: Multi-armed bandits with metric switching costs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part II. LNCS, vol. 5556, pp. 496–507. Springer, Heidelberg (2009)
22. Guha, S., Munagala, K.: Approximation algorithms for bayesian multi-armed bandit problems. CoRR, full version, also subsumes [20, 21] and [23] (2013)
23. Guha, S., Munagala, K., Pál, M.: Iterated allocations with delayed feedback. Manuscript, available at CoRR (2010), `http://arxiv.org/abs/1011.1161`
24. Gupta, A., Krishnaswamy, R., Molinaro, M., Ravi, R.: Approximation algorithms for correlated knapsacks and non-martingale bandits. In: Proc. of FOCS (2011)
25. Jain, K., Vazirani, V.V.: Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. J. ACM 48(2), 274–296 (2001)
26. Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics 6, 4–22 (1985)
27. Lee, J., Sviridenko, M., Vondrk, J.: Submodular maximization over multiple matroids via generalized exchange properties. Math. Oper. Res. 35(4), 795–806 (2010)
28. Ny, J.L., Dahleh, M., Feron, E.: Multi-UAV dynamic routing with partial observations using restless bandits allocation indices. In: Proceedings of the 2008 American Control Conference (2008)
29. Simon, R.: Adaptive treatment assignment methods and clinical trials. Biometrics 33(4), 743–749 (1977)
30. Streeter, M.J., Golovin, D.: An online algorithm for maximizing submodular functions. In: NIPS, pp. 1577–1584 (2008)
31. Streeter, M.J., Smith, S.F.: An asymptotically optimal algorithm for the max k-armed bandit problem. In: Proceedings of AAAI (2006)
32. Suzuki, Y.: On sequential decision problems with delayed observations. Annals of the Institute of Statistical Mathematics 18(1), 229–267 (1966)
33. Tian, F., DeWitt, D.J.: Tuple routing strategies for distributed eddies. In: Proc. of VLDB (2003)

The Approximability of the Binary Paintshop Problem

Anupam Gupta[1,⋆], Satyen Kale[2], Viswanath Nagarajan[2], Rishi Saket[2], and Baruch Schieber[2]

[1] Dept. of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213
anupamg@cs.cmu.edu
[2] IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598
{sckale,viswanath,rsaket,sbar}@us.ibm.com

Abstract. In the Binary Paintshop problem, there are m cars appearing in a sequence of length $2m$, with each car occurring twice. Each car needs to be colored with two colors. The goal is to choose for each car, which of its occurrences receives either color, so as to minimize the total number of color changes in the sequence. We show that the Binary Paintshop problem is equivalent (up to constant factors) to the Minimum Uncut problem, under randomized reductions. By derandomizing this reduction for hard instances of the Min Uncut problem arising from the Unique Games Conjecture, we show that the Binary Paintshop problem is $\omega(1)$-hard to approximate (assuming the UGC). This answers an open question from [BEH06,MS09,AH11].

1 Introduction

The paintshop problem is defined as follows: we are given a $2m$ length sequence containing m cars, where each car appears twice. Each car need to be colored red in one occurrence, and blue in the other. We need to choose which occurrence for each car to color with which color — the goal is to minimize the number of times we need to change the current color. E.g., for $m = 3$, we may represent the 3 cars by x, y, z. If the sequence is $x_1x_2y_1z_1y_2z_2$, where the subscripts denote the first and second occurrence of each car, we could use the colors BRRRBB, to get two color changes, which is the minimum possible. This problem (along with generalizations) was introduced by Epping, Hottstättler and Oertel [EHO01]; their motivation was a natural application in the automotive industry.

Let us formalize the definition: in the basic *Binary Paintshop* problem, the input is a sequence of length n (which is usually associated with the set $[n] := \{1, 2, \ldots, n\}$), along with a matching H on the points in $[n]$. A feasible coloring $f : [n] \to \{\mathsf{B}, \mathsf{R}\}$ of the vertices must ensure that the endpoints of each matching edge $(i, j) \in H$ are bi-colored—i.e., feasibility means $f(i) \neq f(j)$ for all $(i, j) \in$

⋆ This author thanks IBM T.J. Watson Research Center for the generous hospitality while this work was done.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 205–217, 2013.

H. The *cost* of a coloring, also called the number of *color changes* is the number of pairs $(i, i+1)$ for $i \in [n-1]$ that are bichromatic; i.e.,

$$\text{cost}(f) := \sum_{i=1}^{n-1} \mathbf{1}(f(i) \neq f(i+1)) \ .$$

For an instance of the Binary Paintshop problem Γ, we denote by $\mathsf{Opt}(\Gamma)$ the cost of the minimum cost coloring. The goal is to find a feasible coloring f that (approximately) minimizes the cost. We refer to the edges of H as the *constraints* in the instance.

Epping et al. [EHO01], along with defining the problem, gave an exponential-time dynamic programming algorithm to solve the problem exactly, and showed NP-hardness for it as well. Subsequently, Bonsma et al. [BEH06] and Meunier and Sebö [MS09] showed the problem to be APX-hard. They posed the question of whether the problem admitted a constant-factor approximation algorithm. We resolve this question negatively assuming the Unique Games Conjecture (UGC) of Khot [Kho02]:

Theorem 1. *Assuming the UGC, the (basic) Binary Paintshop problem is NP-hard to approximate within any constant factor.*

The above theorem follows via reduction from *Min Uncut* and is proved in Section 4. The Min Uncut problem is the complement of Maximum Cut, and is defined formally in Section 1.2. We present an approximation preserving (up to constant factors) reduction from Min Uncut to Binary Paintshop in Section 3.1. Assuming the Unique Games Conjecture, Khot et al. [KKMO07] showed that Min Uncut is NP-hard to approximate within any constant factor. Using their result and a specific instantiation of the reduction in Section 3.1 yields the proof of Theorem 1. A connection between the Binary Paintshop and Min Uncut problems was also noted by Meunier and Sebö [MS09], but they could only show an APX-hardness using this connection.

We also consider a generalization of this problem: in the *generalized Binary Paintshop* problem instead of being given a sequence of n points (which is naturally associated with a path graph on n nodes), we are given a general graph $G = (V, E)$. Moreover, instead of the constraints H forming a matching on V, we have a bipartite graph $H = (V, E_H)$ on V. The feasibility of a coloring $f : V \to \{\mathsf{B}, \mathsf{R}\}$ still requires that all constraint edges in H are bi-colored—i.e., $f(i) \neq f(j)$ for all $(i, j) \in E_H$. Note that the bipartiteness of H is essential to ensure feasibility, if H is not bipartite there is no feasible coloring. The interesting cases of this problem are when the constraint graph H has many components—e.g., when H is a matching and hence potentially has $n/2$ components.[1] The cost of f is defined naturally as

$$\text{cost}(f) := \sum_{(i,j) \in E(G)} \mathbf{1}(f(i) \neq f(j)) \ .$$

[1] This is because the problem can be solved in time exponential in the number of connected components of H (and polynomial in $n = |V(G)|$) by enumerating over the 2-colorings of each component of H.

Our second result, proved in Section 3.2, is that the generalized Binary Paintshop problem is no harder than the Min Uncut problem:

Theorem 2. *A ρ-approximation algorithm for Min Uncut can be used to give a ρ-approximation for the (generalized) Binary Paintshop problem.*

Using the algorithm of Agarwal et al. [ACMM05], we now get an $O(\sqrt{\log n})$-approximation for Binary Paintshop. Note that the hardness result is shown for the most restrictive (basic) Binary Paintshop problem, whereas the algorithm is for the generalized Binary Paintshop problem.

1.1 Related Work

Other than considering the complexity of the problem as mentioned above, previous work analyzed the performance of greedy algorithms for the paintshop problem, since this type of algorithms are actually used in real life instances of the problem. Meunier and Sebö [MS09] showed a class of instances for which the greedy algorithm is optimal. Amini et al. [AMMM10] showed that the greedy algorithm is optimal for even a larger class of instances and also proved that the expected number of color changes given by the greedy coloring on a random sequence is at most $2n/3$. Andres and Hottstättler [AH11] improved this upper bound to $n/2$. They also considered a hybrid greedy algorithm whose expected number of color changes is $2n/5$.

1.2 Notation and Definitions

In the *Min Uncut* problem, the input is an undirected, unweighted graph $G = (V, E)$ on $n := |V|$ vertices, with every vertex having degree at most $\text{poly}(n)$ (thus we allow parallel edges). The goal is to find a cut $(U, \overline{U})$ for $U \subseteq V$ to minimize the number of uncut edges—i.e., the edges in $E[U] \cup E[\overline{U}]$, where $E[S]$ is the set of edges both of whose edges lie within the set S. With some abuse of notation, for an instance of the Min Uncut problem G, we denote by $\mathsf{Opt}(G)$ the optimal value. The current best algorithm for Min Uncut is an $O(\sqrt{\log n})$-approximation due to Agarwal et al. [ACMM05].

2 Dispersive Permutations

For our hardness proofs, we will need that random permutations satisfy a certain property, which we call dispersion. In this section, we record this proof.

Definition 1. *Given a coloring $f : [m] \to \{\mathsf{B}, \mathsf{R}\}$, the* complement *coloring $\overline{f}$ is obtained by switching the assignments of f from B to R and vice versa. The coloring f is called* M-non-monochromatic *for $M = \min\left\{|f^{-1}(\mathsf{B})|, |f^{-1}(\mathsf{R})|\right\}$.*

Definition 2. *Given a coloring $f : [m] \to \{\mathsf{B}, \mathsf{R}\}$ that is M-non-monochromatic, a permutation $\sigma : [m] \to [m]$ is called* dispersive *for f if either f or $(\overline{f} \circ \sigma)$ has at least $M/16$ color changes. If a permutation is dispersive for all colorings, then it is simply called* dispersive.

Lemma 1. *For any $m \geq 10^8$, a uniformly random permutation $\sigma : [m] \to [m]$ is dispersive with probability at least $1 - \frac{1}{512m}$.*

Proof. Fix any $M \in [m/2]$ and let f be an M-non-monochromatic coloring. Observe that when $M < 16$, the coloring f trivially has at least $M/16$ color changes; so we assume $M \geq 16$ below.

We first count the number of colorings with at most $M/16$ color changes. For any number $r \leq M/16$, to estimate the number of colorings with exactly r color changes, we note that any such coloring f gives rise to a subset of r indices $\{i_1, i_2, \ldots, i_r\} \subseteq [m-1]$ where the coloring changes color, i.e. for each such i_k, we have $f(i_k) \neq f(i_k + 1)$. The number of such subsets is $\binom{m-1}{r}$, and each such subset can give rise to two colorings with r-color changes. Hence, the number of colorings with at most $M/16$ color changes is bounded by

$$N \;:=\; \sum_{r=0}^{M/16} 2\binom{m-1}{r} \;\leq\; \frac{M}{8}\binom{m-1}{M/16}.$$

Since for a uniformly random permutation σ, $(\overline{f} \circ \sigma)$ is a uniformly random M-non-monochromatic coloring with the opposite minority color of f, of which there are $\binom{m}{M}$, the probability that $(\overline{f} \circ \sigma)$ has at most $M/16$ color changes is bounded by

$$p \quad := \quad \sum_{r=0}^{M/16} \left[\frac{2\binom{m-1}{r}}{\binom{m}{M}} \right] \quad \leq \quad \frac{M\binom{m-1}{M/16}}{8\binom{m}{M}}. \tag{1}$$

Thus, for any fixed M-non-monochromatic coloring f, at most p fraction of permutations σ yield a "bad" coloring $(\overline{f} \circ \sigma)$, i.e. $(\overline{f} \circ \sigma)$ has at most $M/16$ color changes. However, we are only concerned with colorings f which are "bad" to begin with, i.e. f has at most $M/16$ color changes. As above, the number of such colorings is bounded by N. Thus, by the union bound, the probability that a random permutation σ *is not* dispersive for some coloring f that is M-non-monochromatic is at most

$$Np \;\leq\; \frac{M^2\binom{m-1}{M/16}^2}{64\binom{m}{M}} \;\leq\; \frac{(m/2)^2}{64m^4} \;\leq\; \frac{1}{256m^2}, \tag{2}$$

where the second inequality uses Claim 3 below. Summing this probability bound for all $m/2$ integer values for M, we complete the proof of the lemma.

Claim 3. *Suppose $m \geq 10^8$. For any integers $16 \leq M \leq \frac{m}{2}$, we have $\binom{m-1}{M/16}^2 \leq \frac{1}{m^4} \cdot \binom{m}{M}$.*

Proof. The proof is by the following two calculations.

Suppose $M \geq 32 \ln m$. We have $\binom{m-1}{M/16}^2 \leq \left(\frac{16em}{M}\right)^{M/8}$ and $\binom{m}{M} \geq \left(\frac{m}{M}\right)^M$. So the ratio of these is at most:

$$\left(\frac{16em}{M}\right)^{M/8} \cdot \left(\frac{M}{m}\right)^M \;=\; \left(16e(M/m)^7\right)^{M/8} \;\leq\; \left(16e(1/2)^7\right)^{M/8} \;\leq\; e^{-M/8} \;\leq\; \frac{1}{m^4},$$

where the first inequality follows because $M \leq m/2$, and the last because $M \geq 32 \ln m$.

Now suppose $M \leq 32 \ln m$. We have $\binom{m-1}{M/16}^2 \leq m^{M/8}$ and $\binom{m}{M} \geq \left(\frac{m}{M}\right)^M$. Thus the ratio is at most

$$m^{M/8} \cdot \frac{M^M}{m^M} \leq \frac{m^{M/2}}{m^M} \leq \frac{1}{m^4},$$

where the first inequality follows because for $m \geq 10^8$, we have $M \leq 32 \ln(m) \leq m^{3/8}$, and the second because $M \geq 8$.

3 Relationship to Min Uncut

In this section we show formal connections between the Min Uncut problem and the Binary Paintshop problem. Recall the Min Uncut problem from Section 1.2, in which we want to find a cut that minimizes the number of uncut edges. We show that the two problems have the same asymptotic approximability under randomized reductions.

3.1 Reducing Min Uncut to Binary Paintshop

Theorem 4. *A ρ-approximation algorithm for Binary Paintshop implies a randomized algorithm for Min Uncut that returns an $O(\rho)$-approximation with probability at least* 0.99.

The success probability can be boosted to arbitrarily close to 1 by repeating the algorithm with different random seeds and returning the best solution found.

The proof works as follows. Given an instance of the Min Uncut problem, we give a gadget transformation to an instance of the Binary Paintshop problem. This gadget has a block of nodes for each node in the Min Uncut instance, and matchings between different blocks represent edges in the Min Uncut instance. Since a solution to the Binary Paintshop instance is forced to color the endpoints of each matching edge differently, within every block we can interpret the different colors assigned as "votes" for the side of the cut that the node in the Min Uncut instance should lie on. One can then round this solution to the Binary Paintshop instance to a solution for the Min Uncut instance by taking the majority vote within each block. To ensure that the cost of the obtained solution to the Min Uncut can be bounded in terms of the cost of the solution to the Binary Paintshop instance, we need to relate the minority vote in each block to the number of color changes. To do this, we add an additional block for each node together with a matching provided by a dispersive permutation into the original block which serves to "mix up" the coloring of the original block. This ensures that the total number of the color changes within the two blocks for each node is at least a constant fraction of the minority vote. The details follow.

The Binary Paintshop Instance Γ_G. We are given a graph $G(V, E)$ as input to the Min Uncut problem. Let $n = |V|$ where vertices are indexed $\{1, 2, \ldots, n\}$. Let $d(i)$ denote the degree of vertex $i \in [n]$. We choose an integer parameter $T \leq poly(n)$ to be specified later, and consider the multigraph G' obtained by making T copies of each edge in G, so each vertex i has degree $Td(i)$ in G', and we order the corresponding edges arbitrarily. For each $i \in [n]$, we choose a random permutation σ_i on $Td(i)$ elements. Our instance Γ_G of Binary Paintshop contains for each vertex $i \in [n]$, two *sequences* of points R_i and S_i. (See Figure 1.)

Sequence R_i: This contains $2Td(i)$ points and is given by

$$\langle x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2}, \cdots, x_{i,Td(i)}, y_{i,Td(i)} \rangle.$$

There are $Td(i)$ x-points corresponding to edges incident to vertex i in G', and the remaining are y-points which will be used to enforce a feasible coloring.

Sequence S_i: This contains $Td(i)$ points

$$\langle z_{i,1}, z_{i,2}, \cdots, z_{i,Td(i)} \rangle.$$

Final sequence W: The final sequence W of the instance Γ_G is just a concatenation of the sequences constructed above.

$$W := R_1 \circ R_2 \circ \cdots \circ R_n \circ S_1 \circ S_2 \circ \cdots \circ S_n. \tag{3}$$

The instance Γ_G of Binary Paintshop now consists of the path whose vertices correspond to the points in W, in that order. Note the length of this path is $3T \sum_{i=1}^{n} d(i) = 6T|E|$, which is polynomial in the size of G. Now for the constraints in Γ_G—recall these must form a matching H on the points in W. There are two kinds of matching pairs: *edge pairs* and *permutation pairs*, which are defined below.

Edge pairs: For each edge e in G', if e is the rth edge incident to vertex i and the sth edge incident to vertex j, then we define $\{x_{i,r}, x_{j,s}\}$ to be an edge pair.

Permutation pairs: For each vertex $i \in [n]$ and each $\ell \in \{1, 2, \cdots, Td(i)\}$, we have a permutation pair $\{y_{i,\ell}, z_{i,\sigma_i(\ell)}\}$, where σ_is are the random permutations chosen above.

Relating $\mathsf{Opt}(\Gamma_G)$ and $\mathsf{Opt}(G)$. We now relate the optimal solutions on any Min Uncut instance G, and the Binary Paintshop instance Γ_G created by the above process.

Lemma 2. *Given any feasible solution to the Min Uncut problem on G with M uncut edges, we can construct a feasible solution to the Binary Paintshop instance Γ_G of cost at most $2MT + 2n$. Thus $\mathsf{Opt}(\Gamma_G) \leq 2T \cdot \mathsf{Opt}(G) + 2n$.*

Proof. Let $(U, \overline{U})$ be a cut in G with M uncut edges. We now construct a feasible solution to the Paintshop instance Γ_G of cost at most $2MT + 2n$. We first construct an initial coloring $F_{initial}$ of the points in Γ_G as follows. (This initial

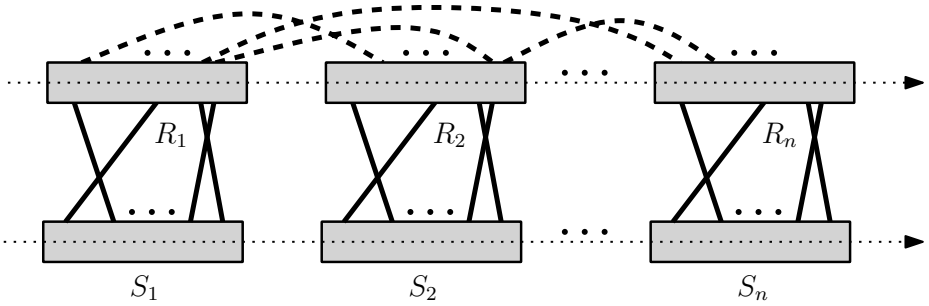


Fig. 1. High level view of the construction. The rectangles represent the sequences R_i and S_i, and the dotted line represents their concatenation in the final sequence W. The edge pairs are represented by the dashed lines, whereas the permutation pairs are solid lines.

coloring will be infeasible, but we shall see that incurring a small extra cost will make it feasible.) For each $1 \leq i \leq n$, the points in the sequence R_i are colored R if $i \in U$ and B otherwise. Similarly, all points in the sequence S_i are colored B if $i \in U$ and R otherwise, for $1 \leq i \leq n$. (In Figure 1, all points within each rectangle have the same color, and any pair of vertically aligned rectangles has opposite colors.) Note that every permutation pair is bichromatic and hence satisfied. Before we fix the monochromatic edge pairs, let us first count the number of color changes in $F_{initial}$. Clearly, since each of the sequences S_i and R_i are monochromatic, the only color changes are between adjacent sequences of the form R_i and R_{i+1}, S_i and S_{i+1}, or R_n and S_1. So the total number of color changes in $F_{initial}$ is at most $2n$.

However, as mentioned above, some of the edge pairs may be monochromatic. If an edge $e = \{i, j\}$ is not separated by the cut $(U, \overline{U})$, then all T pairs, corresponding to the copies of this edge in G', are monochromatic. To fix this, we flip the color of one of the endpoints in every such pair. Call this new coloring F^*, which is feasible by construction. Observe that this process incurs an additional cost of at most $2T$ for each uncut edge in G, since flipping the color of a point may make both its neighboring edges in W two-colored. Since the total number of uncut edges in the Min Uncut instance is M, and handling each one incurs an extra cost of at most $2T$, the number of color changes in F^* is at most $2MT$ greater than that in $F_{initial}$, and hence at most $2MT + 2n$.

Lemma 3. *Suppose all the permutations σ_i chosen in the reduction are dispersive. Then given a feasible coloring for Γ_G with C color changes, we can construct a feasible solution to minimum uncut on G with cost at most $33\,(C/T)$. Thus we have $\mathsf{Opt}(G) \leq \frac{33}{T} \cdot \mathsf{Opt}(\Gamma_G)$.*

Proof. Consider some feasible coloring F for Γ_G, with cost (a.k.a. number of color changes) C. We will construct a feasible solution to minimum uncut on G where the number of uncut edges is at most $O(C/T)$.

For each $i \in [n]$, define the *majority color* of i (under the feasible coloring F for Γ_G) to be the one that is represented at least $|R_i|/2$ times among the points in R_i; the *minority color* of i is defined to be the opposite color. Also, for $i \in [n]$, let $q_i \leq |R_i|/2$ denote the number of points in R_i colored with the minority color. Let $U \subseteq [n]$ denote the vertices i having majority color B, and hence $\overline{U}$ is the set of vertices having majority color R. In the rest of the proof, we show that the cost of the solution $(U, \overline{U})$ to Min Uncut is $D \leq 33 \cdot \frac{C}{T}$.

Consider any uncut edge (u, v) in this solution $(U, \overline{U})$. There are T edge-pairs between points of R_u and R_v. By the feasibility of F, these must be colored with opposite colors. Since u and v are either both in U or both in $\overline{U}$, they have the same majority color: so the $2T$ points in $R_u \cup R_v$ corresponding to edge (u, v) contribute at least T minority colors at u and v. Thus the total number of points colored with the minority color is $\sum_{i=1}^{n} q_i \geq T \cdot D$. Rearranging,

$$D \quad \leq \quad \frac{1}{T} \sum_{i=1}^{n} q_i. \tag{4}$$

Fix any $i \in [n]$. Since q_i equals the number of minority colors in R_i, by Definition 1, the coloring $F[R_i]$ is q_i-non-monochromatic. Now we can use the dispersive property of the permutation σ_i to prove the following claim.

Claim 5. *For any $i \in [n]$, let the total number of color changes in R_i and S_i be C_i. Then $q_i \leq 33\, C_i$.*

Proof. Let $R_i := X_i \cup Y_i$ where X_i and Y_i consist of the x-points and y-points, respectively. Let $F[Y_i]$ be the coloring F restricted to the subsequence of the y-points. Say the minority color in $F[Y_i]$ is B, and let B_y denote the number of B-colored points in Y_i. This implies that $F[Y_i]$ is B_y-non-monochromatic.

We first claim that $B_y \leq 16C_i$. Suppose (for a contradiction) that $B_y > 16C_i$. Then since σ_i is dispersive, either $F[Y_i]$ or $\overline{F}[Y_i \circ \sigma_i]$ has at least $B_y/16 > C_i$ color changes. Note that $\overline{F}[Y_i \circ \sigma_i]$ is precisely the coloring $F[S_i]$ since the points in Y_i and S_i are paired (under permutation σ_i) and so have opposite colors by the feasibility of F. Note also that the number of color changes in $F[R_i]$ is at least that number in $F[Y_i]$ since Y_i is a subsequence of R_i. It follows that the number of color changes in $F[R_i]$ and $F[S_i]$ is greater than C_i, giving us a contradiction. Thus we must have $B_y \leq 16C_i$.

Finally, the sequence R_i is obtained by alternating between X_i and Y_i. Since the number of color changes in $F[R_i]$ is at most C_i, the number of B-colored points in $F[X_i]$ and $F[Y_i]$ must differ by at most C_i. The latter quantity is at most $16C_i$ by the previous argument, so the number of B-colored points in $F[X_i]$ is at most $17C_i$. It follows that the total number of B-colored points in R_i is at most $33C_i$, and hence $q_i \leq 33C_i$.

Using the bound from Claim 5 for each $i \in [n]$, along with (4),

$$D \quad \leq \quad \frac{1}{T} \sum_{i=1}^{n} q_i \quad \leq \quad \frac{33}{T} \sum_{i=1}^{n} C_i \quad \leq \quad 33 \frac{C}{T}.$$

This completes the proof of Lemma 3.

Completing the proof of Theorem 4. Having related the optima for the Min Uncut instance G and the Binary Paintshop instance Γ_G, we can now prove Theorem 4.

Proof. We may assume that $\mathsf{Opt}(G) \geq 1$: the case $\mathsf{Opt}(G) = 0$ corresponds exactly to checking that G is bipartite and this can be easily done in polynomial time. By the assumption of the theorem, there is a ρ-approximation algorithm for Binary Paintshop, where $\rho \geq 1$. Choosing $T = \max\{10^8, n\}$ means that the probability of some fixed permutation chosen in the gadget not being dispersive is at most $\frac{1}{512n}$; by a union bound, the probability that all permutations chosen are dispersive is at least $1 - 1/512 > 0.99$. If we run the claimed ρ-approximation algorithm for the Binary Paintshop instance Γ_G, we get a feasible coloring for Γ_G with $C \leq \rho\,\mathsf{Opt}(\Gamma_G)$ color changes. By Lemma 2 we know that $\mathsf{Opt}(\Gamma_G) \leq 2T\,\mathsf{Opt}(G) + 2n$. Using the "decoding" algorithm from Lemma 3, we can now construct a feasible solution to the Min Uncut instance G with cost at most

$$33\,\frac{C}{T} \;\leq\; 33\rho\,\frac{\mathsf{Opt}(\Gamma_G)}{T} \;\leq\; 33\rho\left(\frac{2T\,\mathsf{Opt}(G) + 2n}{T}\right) \;\leq\; 132\,\rho\,\mathsf{Opt}(G),$$

since $\mathsf{Opt}(G) \geq 1$ and $T \geq n$. This completes the proof of Theorem 4.

3.2 Reducing Binary Paintshop to Minimum Uncut

We now give a reduction in the opposite direction, showing that the Binary Paintshop problem is essentially a special case of Min Uncut. In fact, we show that even the *generalized* Binary Paintshop problem can be solved using an algorithm for Min Uncut.

Theorem 6. *Given a ρ-approximation algorithm for the Min Uncut problem, we get a ρ-approximation algorithm for generalized Binary Paintshop.*

Proof. Consider an instance of generalized Binary Paintshop: recall that this consists of a graph $G = (V, E_G)$, and the constraint graph $H = (V, E_H)$, and we want to find a coloring that bi-colors each of the edges in E_H while cutting the fewest edges in E_G. Let C denote the cost of the optimal solution to this instance; let $|V_G| = n$ and hence $C \leq |E_G| \leq \binom{n}{2}$.

Let $H_1, H_2, \ldots, H_k$ be the k connected components of $H = (V, E_H)$. Since H is bipartite, it is easy to see that each H_i is bipartite with a unique bipartition of vertices we denote by (V_i^0, V_i^1), $i = 1, \ldots, k$. The instance $I = (U, E_I)$ of Min Uncut is constructed as follows. The vertex set $U := \cup_{i=1}^{k} \{u_i^0, u_i^1\}$. For each $i = 1, \ldots, k$, add $t := \rho n^2$ edges between u_i^0 and u_i^1 in graph I.

Consider an edge $e \in E_G$ whose end points lie in V_i^a and V_j^b for some $i \leq j$ and $a, b \in \{0, 1\}$. Add a corresponding edge in graph I between $u_i^{a'}$ and u_j^b, where $a' = 1 - a$. Note that this may lead to self-loops in I. The number of edges in I is $|E_G| + tk$.

Consider a feasible solution to the Binary Paintshop instance. This solution colors all the nodes in V_i^0 the same color, and the opposite color for all the nodes

in V_i^1, for $i = 1, \ldots, k$. To construct a solution for the Min Uncut instance I, add each u_i^a to one side of the cut according to the color of V_i^a for $i = 1, \ldots, k$ and $a \in \{0, 1\}$. This separates all the edges between u_i^0 and u_i^1. The only edges that may remain uncut are those corresponding to edges in E_G. An edge $e \in E_G$ with end points in V_i^a and V_j^b for some $i \leq j$ and $a, b \in \{0, 1\}$ corresponds to an edge $e' \in E_I$ between $u_i^{a'}$ and u_j^b, where $a' = 1 - a$. Thus, e' is cut in the uncut solution if and only if e was monochromatic in the Binary Paintshop solution. That is, the number of edges in E_I that are not separated (the uncut objective value) is exactly equal to the number of non-monochromatic edges in E_G (the paintshop objective value).

Conversely, any solution to the Min Uncut instance I that separates each pair $\{u_i^0, u_i^1\}$ $(i = 1, \ldots, k)$ can be turned into a feasible solution for the Binary Paintshop instance of the same cost by coloring each V_i^a according to the side of the cut containing u_i^a for $a \in \{0, 1\}$. Now, the output of the ρ-approximation algorithm for Min Uncut on I must separate each pair $\{u_i^0, u_i^1\}$: else, its cost is at least $t = \rho n^2 > \rho C$, contradicting the fact that the output cut is a ρ-approximation.

Combining Theorems 4 and 6, we get that the approximability of these two problems is the same (up to constant factors) under randomized reductions.

Remark. It is easy to extend our results to weighted versions of the Binary Paintshop problem where each adjacent pair in the sequence comes with some cost.

4 UGC Hardness of Approximation

In this section we shall prove the desired Unique Games Conjecture based hardness of approximation for the Binary Paintshop problem via the above connection to the Min Uncut problem. We begin by stating the current best UGC based inapproximability result for Min Uncut. The following theorem is based on UGC for *regular* Unique Games – where the degree is a constant depending on the *completeness* and *soundness* of the instance[2].

Theorem 7 ([KKMO07]). *Assuming the Unique Games Conjecture [Kho02] the following holds. For every constant $\varepsilon > 0$, there is a positive integer $d := d(\varepsilon)$ such that given a d-regular n-vertex graph G as an instance of Min Uncut, it is NP-hard to decide between the following two cases:*

1. *YES Case:* $\mathsf{Opt}(G) \leq \frac{1}{2}\varepsilon nd$.
2. *NO Case:* $\mathsf{Opt}(G) \geq \frac{1}{4}\sqrt{\varepsilon}nd$.

The formal statement of the reduction is given below and, along with Theorem 7, implies Theorem 1.

[2] Applying a pre-processing step of Dinur (Lemma 4.1 of [Din07]) using constant degree expanders followed by Parallel Repetition [Raz98], a general instance of Unique Games can be deterministically converted to a regular instance and thus, one can assume UGC holds for regular instances of Unique Games.

Theorem 8. *There is a polynomial-time (deterministic) reduction from instances G of Min Uncut given by Theorem 7 with parameter $\varepsilon > 0$, to instances Γ of the Binary Paintshop problem on a sequence of length $N = poly(n)$ such that,*

1. *YES Case: If G is a YES instance of Min Uncut then $\mathsf{OPT}(\Gamma) \leq \varepsilon N$*
2. *NO Case: If G is a NO instance of Min Uncut then $\mathsf{OPT}(\Gamma) \geq \frac{1}{400}\sqrt{\varepsilon}N$.*

Proof. We instantiate the reduction of Section 3.1 with $T = \max\{10^8, 1/\varepsilon\}$. Since all nodes have the same degree d, we only require a single dispersive permutation $\sigma : [dT] \to [dT]$. Since dT is a constant (depending on ε), such a permutation can be found using a brute-force search. Note that the length of the Binary Paintshop instance generated is $N = 3ndT$.

Now if G is a YES instance of Min Uncut then Lemma 2 implies that

$$\mathsf{Opt}(\Gamma) \;\leq\; 2T\mathsf{Opt}(G) + 2n \;\leq\; \varepsilon ndT + 2n \;\leq\; \varepsilon N.$$

If G is a NO instance of Min Uncut then Lemma 3 implies that

$$\mathsf{Opt}(\Gamma) \;\geq\; \frac{T}{33}\mathsf{Opt}(G) \;\geq\; \frac{1}{132}\sqrt{\varepsilon}ndT \;\geq\; \frac{1}{400}\sqrt{\varepsilon}N.$$

5 A Ternary Paintshop Problem

A natural extension of the (generalized) Binary Paintshop problem is to the case where there are three or more colors. The goal is again to ensure that the constraint edges are not monochromatic, and to minimize the number of bichromatic edges in G. One natural hurdle in this case is that checking whether there exists a feasible solution becomes NP-hard, since we would have to check whether the given constraint graph H is k-colorable for $k \geq 3$. However, even when the constraints are trivially k-colorable, and the graph G is very simple, we show that the problem remains very hard to approximate.

Specifically, consider the ternary case with $k = 3$ colors, where the underlying graph G is a collection of disjoint paths, and the constraint graph H is a matching. We show it is NP-hard to identify whether the optimal cost is zero or not, and hence NP-hard to approximate to any factor. Indeed, take a graph $G_c = (V_c, E_c)$ that is an instance of 3-coloring, and construct an instance (G, H) of ternary paintshop as follows: for each vertex $v \in V_c$, construct $\delta(v)$ vertices in $V(G)$, with one copy corresponding to each edge $e \in E_c$ incident to v, and connect all these $\delta(v)$ copies by a path P_v. These $\sum_{v \in V_c} \delta(v) = 2|E_c|$ vertices and $\sum_{v \in V_c}(\delta(v) - 1) = 2|E_c| - |V_c|$ edges form the graph G. Now for each edge $e = (u, v) \in E_c$, add a constraint edge in H between the corresponding copies of u and v in $V(G)$ — hence the constraint edges H form a matching. Now G_c has a valid 3-coloring if and only if the ternary paintshop instance (G, H) has a solution that monochromatically colors each of the $|V_c|$ paths and cuts zero edges.

In the above reduction we crucially used the fact that G was a forest, and had disconnected components. This can be remedied to show a slightly weaker hardness result. Define the *(basic) ternary paintshop* problem as follows: given a sequence of length n (again associated with the integers $[n] = \{1, 2, \ldots, n\}$), and a matching H on the points $[n]$, find a coloring f that ensures that each edge in H is bichromatic, and minimizes the number of bichromatic pairs $(i, i+1)$.

Theorem 9. *For any constant $\varepsilon > 0$, the basic ternary paintshop problem is NP-hard to approximate to within $n^{1-\varepsilon}$ in polynomial time.*

Proof. Consider the same reduction from a 3-coloring instance $G_c = (V_c, E_c)$ to the ternary paintshop instance (G, H) on disconnected paths, but now take T copies of the graphs (G, H). Obtain a sequence of length $N := T \cdot |V(G)|$ by considering the vertices of V_c in some order, and laying down all the paths P_v for the same vertex in each of these T copies consecutively; the constraints are inherited from the original instances (G, H). This gives the instance Γ for the basic ternary paintshop problem with sequence length N. Note that a 3-coloring for G_c naturally gives a ternary coloring of $[N]$ with at most $|V_c| - 1$ many color changes. On the other hand, if G_c is not 3-colorable, each of the T instances of (G, H) must incur at least one color change, and the number of color changes in Γ is at least T. If $n := |V_c|$, then setting $T = n^{2/\varepsilon}$ means it is NP-hard to distinguish the case when the optimum is at most $n \approx N^{\varepsilon}$ and when it is at least $T \approx N^{1-\varepsilon}$, giving us the claimed hardness.

A different version of the ternary paintshop problem (e.g., from [MS09]) is where the constraints are hyperedges of size 3, and also form a matching — i.e., none of the hyperedges share vertices from $[n]$. A constraint $\{i, j, k\}$ now means the three vertices i, j, k must be given distinct colors. The reduction from Theorem 9 easily extends to show hardness for this variant too: for each constraint $e = \{i, j\}$ in that reduction, add a new dummy vertex v_e and use the constraint $\{i, j, v_e\}$. Other extensions considered in previous papers, with constraints of the form *"the set $S \subseteq [n]$ must contain exactly t_{iS} nodes of color i for each color $i \in [k]$, where $\sum_{i \in [k]} t_{iS} = |S|$"*, are thus at least as hard.

References

ACMM05. Agarwal, A., Charikar, M., Makarychev, K., Makarychev, Y.: $O(\sqrt{\log n})$ approximation algorithms for min UnCut, min 2CNF deletion, and directed cut problems. In: 37th Annual ACM Symposium on Theory of Computing, pp. 573–581 (2005)

AH11. Andres, S.D., Hochstättler, W.: Some heuristics for the binary paint shop problem and their expected number of colour changes. J. Discrete Algorithms 9(2), 203–211 (2011)

AMMM10. Amini, H., Meunier, F., Michel, H., Mohajeri, A.: Greedy colorings for the binary paintshop problem. Journal of Discrete Algorithms 8(1), 8–14 (2010)

BEH06. Bonsma, P.S., Epping, T., Hochstättler, W.: Complexity results on restricted instances of a paint shop problem for words. Discrete Applied Mathematics 154(9), 1335–1343 (2006)

Din07. Dinur, I.: The PCP theorem by gap amplification. Journal of the ACM 54(3), 12 (2007)

EHO01. Epping, T., Hochstättler, W., Oertel, P.: Some results on a Paint Shop problem for words. Electronic Notes in Discrete Mathematics 8, 31–33 (2001)

Kho02. Khot, S.: On the power of unique 2-prover 1-round games. In: 34th Annual ACM Symposium on the Theory of Computing, pp. 767–775 (July 2002)

KKMO07. Khot, S., Kindler, G., Mossel, E., O'Donnell, R.: Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? SIAM J. Comput. 37(1), 319–357 (2007)

MS09. Meunier, F., Sebö, A.: Paintshop, odd cycles and necklace splitting. Discrete Applied Mathematics 157(4), 780–793 (2009)

Raz98. Raz, R.: A parallel repetition theorem. SIAM Journal of Computing 27(3), 763–803 (1998)

Approximation Algorithms for Movement Repairmen

MohammadTaghi Hajiaghayi[1,⋆], Rohit Khandekar[2], M. Reza Khani[3,⋆], and Guy Kortsarz[4,⋆⋆]

[1] University of Maryland, College Park, MD, and AT&T Labs, Florham Park, NJ
hajiagha@cs.umd.edu
[2] Knight Capital Group, Jersey City, NJ
rkhandekar@gmail.com
[3] University of Maryland, College Park, MD
khani@cs.umd.edu
[4] Rutgers University, Camden, NJ
guyk@camden.rutgers.edu

Abstract. In the *Movement Repairmen (MR)* problem we are given a metric space (V, d) along with a set R of k repairmen $r_1, r_2, \ldots, r_k$ with their start depots $s_1, s_2, \ldots, s_k \in V$ and speeds $v_1, v_2, \ldots, v_k \geq 0$ respectively and a set C of m clients $c_1, c_2, \ldots, c_m$ having start locations $s'_1, s'_2, \ldots, s'_m \in V$ and speeds $v'_1, v'_2, \ldots, v'_m \geq 0$ respectively. If t is the earliest time a client c_j is collocated with any repairman (say, r_i) at a node u, we say that the client is served by r_i at u and that its latency is t. The objective in the (Sum-MR) problem is to plan the movements for all repairmen and clients to minimize the sum (average) of the clients latencies. The motivation for this problem comes, for example, from Amazon Locker Delivery [Ama10] and USPS gopost [Ser10]. We give the first $O(\log n)$-approximation algorithm for the Sum-MR problem. In order to solve Sum-MR we formulate an LP for the problem and bound its integrality gap. Our LP has exponentially many variables, therefore we need a separation oracle for the dual LP. This separation oracle is an instance of Neighborhood Prize Collecting Steiner Tree (NPCST) problem in which we want to find a tree with weight at most L collecting the maximum profit from the clients by visiting at least one node from their neighborhoods. The NPCST problem, even with the possibility to violate both the tree weight and neighborhood radii, is still very hard to approximate. We deal with this difficulty by using LP with geometrically increasing segments of the time line, and by giving a *tricriteria* approximation for the problem. The rounding needs a relatively involved analysis. We give a constant approximation algorithm for Sum-MR in Euclidean Space where the speed of the clients differ by a constant factor. We also give a constant approximation for the makespan variant.

⋆ Supported in part by NSF CAREER award 1053605, NSF grant CCF-1161626, ONR YIP award N000141110662, DARPA/AFOSR grant FA9550-12-1-0423, and a University of Maryland Research and Scholarship Award (RASA).
⋆⋆ Supported in part by NSF award number 434923.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 218–232, 2013.

1 Introduction

In the well-known *Traveling Repairman (TR)* problem, the goal is to find a tour to cover a set of clients such that the sum of latencies seen by the clients is minimized. The problem is also known as the *minimum latency problem*, see [GK98], the *School-bus driver problem*, see [WAW93] etc. This problem is well studied in the operations research literature and has lots of applications in real world, see for example [BYCR93]. The problem is NP-Hard even in tree metrics [Sit06]. Blum et al. [BCC$^+$94] give the first constant-factor approximation algorithm for the TR problem. They also observe that there is no PTAS ($(1+\epsilon)$-approximation algorithm for an arbitrary constant $\epsilon > 0$) for the problem unless $P = NP$. After a sequence of improvements, Chaudhuri et al. [CGRT03] give a 3.59-approximation algorithm for TR which is the current best approximation factor for this problem.

Fakcharoenphol et al. [FHR03] generalize the TR problem to the k-Traveling Repairman (k-TR) problem in which instead of one repairman, we can use k repairmen to service the clients where all the repairmen start from the same depot. They give a 16.994-approximation algorithm for k-TR. Chekuri and Kumar [CK04] give a 24-approximation algorithm for the "multiple-depot" version of the k-TR problem where the repairmen can start from different depots.

Chakrabarty and Swamy [CS11] give a constant-factor approximation algorithm for the classical TR problem by introducing two new LPs. Their work is significant as it is the first LP approach to solve the problem.

We generalize [CS11] for the k-TR problem by allowing the repairmen to start from different starting depots and to have different speeds. More importantly, we give the clients ability to move with different speeds, which makes the problem significantly harder. We formally define Sum-MR as follows.

Definition 1. *In the* Sum-MR *problem the inputs are given as follow.*

- *A metric space $\mathcal{M} = (V, d)$ where V is the set of nodes and $d : (V \times V) \to \mathbb{Q}^+$ is the distance function.*
- *A set R of k repairmen $r_1, r_2, \ldots, r_k$. Each repairman r_i has a start depot $s_i \in V$ and speed $v_i \in \mathbb{Q}^+$.*
- *A set C of m clients $c_1, c_2, \ldots, c_m$. Each client c_j has a start location $s'_j \in V$ and speed $v'_j \in \mathbb{Q}^+$.*

A solution to the problem consists of the following.

- *A pair (u_j, t_j) for each client c_j such that c_j can reach node u_j by time t_j considering its speed v'_j (i.e., $d(s'_j, u_j) \leq v'_j \cdot t_j$).*
- *A path p_i for each repairman r_i. In general p_i may not be a simple path and can contain a node or an edge multiple times. Repairman r_i can travel along p_i with maximum speed v_i.*
- *For each pair (u_j, t_j) assigned to client c_j there has to be at least one repairman (r_i) such that r_i visits u_j at time t_j when it travels path p_i.*

The objective for Sum-MR *is to minimize $\sum_{j=1}^{m} t_j$.*

The problem is very natural and is also motivated by the following real-world scenario. Amazon Locker Delivery is an optional shipping method in Amazon online stores. In this method clients have an option to select a certain locker location to pick up their purchased items. Afterwards, Amazon puts the items into a locker in the specified location and sends the locker number and its key combination, to the customer. The package can be picked up by the customer who can go to the locker location by her own means. A very similar delivery option is also offered by the United States Postal Service which is known as gopost [Ser10].

Our algorithm can be used directly in order to plan the movements to minimize the average latency (or the maximum latency). Here the locations of the Amazon stores, clients' homes and locker locations can be thought as the nodes in the metric space in Sum-MR and the repairmen are the shipping vehicles starting from the Amazon stores with different speeds. Moreover, we can take as input how customers are going to pick up their packages (*e.g.* by a car, public transport, bike, and etc.) which realizes the different speeds for the clients. Note that unlike Sum-MR in this scenario it is not necessary for both a repairman and a client to meet at the same node and the same time in order to serve; but if a repairman visits a node at time t a client can visit the node at any time after t and still get served. We formalize these methods of serving and show that the difference in the objective of Sum-MR for the two methods is at most $3 + \epsilon$ in Section 3.1.

In Section 2 we give the outlines of our techniques and summarize all our results. Section 3 contains the detailed explanation of our method to solve Sum-MR in three subsections. Subsection 3.1 contains the necessary preliminaries, in Subsection 3.2 we give our LP formulation, show how to solve it in Subsection 3.3, and finally in Subsection 3.4 we show how to round a fractional solution to the LP to get an integral solution to the Sum-MR.

We describe in detail connection of Sum-MR to the movement framework, neighborhood TSP problems, and orienteering problems in the full version of this paper [HKKK13]. It turns out that the separation oracle for our LP is a generalization of the Neighborhood TSP problem, we give the results related to the separation oracle problem in the full version of this paper [HKKK13]. We also give the formal proof of our results for Euclidean space and the materials related to the Max-MR problem in the full version.

2 Results and Techniques

In this section we summarize all our results along with the overview of their proofs. All the ideas explained here are new in this context.

Our main result is an $O(\log n)$-approximation algorithm for Sum-MR. More precisely we prove the following theorem.

Theorem 1. *There is an $O(\log n)$-approximation algorithm for the* Sum-MR *problem which also upper bounds the integrality gap of its LP formulation.*

We present the novel properties (in this context) of our techniques. First we relax conditions on serving the clients. If a client collocates with a repairman at a certain node of the metric space during the movements we say it gets served *perfectly*. On the other hand if a client visits a node through which a repairman has passed no later than the arrival of the client, we say it gets served *indirectly*. We design a procedure that transforms any solution to the SUM-MR problem where the clients are served indirectly to a solution where all the clients are served perfectly by increasing the total latency with a multiplicative factor at most $3+\epsilon$. We solve SUM-MR for the case when we serve the clients indirectly and use the procedure to serve the clients perfectly. We give an LP formulation for SUM-MR (to serve the clients indirectly) and bound its integrality gap. However there are two major challenges in order to do so. First, solving the LP which has exponentially many variables and second, rounding a solution to the LP efficiently to an integral solution.

In order to solve the LP we need a separation oracle for its dual which turns out to be the following problem.

Definition 2. *Neighborhood Prize Collecting Steiner Tree (NPCST): An instance of the NPCST problem consist of an ordered tuple (V, d, r, C, L) where V is the set of nodes, d is a metric distance function on the set V, $r \in V$ is the root node, C is the set of clients, and L is the cost budget. Each client $c \in C$ is associated with a profit θ_c and a neighborhood ball $\mathcal{B}(c, t_c)$ which contains all the nodes u with $d(u, c) \leq t_c$. The goal is to find a Steiner tree T_{OPT} such that cost$(T_{\mathrm{OPT}}) \leq L$ and the sum of the profits of the clients whose $\mathcal{B}$-ball hits T_{OPT} is maximized.*

The vehicle routing problems become significantly harder when instead of visiting a node, it is sufficient to visit a neighborhood around it. For example in the Neighborhood Steiner Tree (NST) problem, we are given a graph G with a set of clients C where each client c is associated with a neighborhood ball. The objective for NST is to find a tree T with minimum weight that serves at least one node from each client's neighborhood ball. We will prove the following hardness result about the NST problem which shows the source of difficulty in our problem.

Theorem 2. *There is no $O(\log^{2-\epsilon} n)$-approximation algorithm for the NST problem unless NP has quasi-polynomial Las-Vegas algorithms.*

To avoid this hardness we allow relaxing the NPCST constraints. More formally we accept a tri-criteria approximation algorithm for NPCST as our separation oracle defined formally below.

Definition 3. *A (σ, ϕ, ω)-approximation algorithm for the instance (V, d, r, C, L) of the NPCST problem finds a Steiner tree T with the following properties; T is said to hit a client c with $\mathcal{B}$-ball $\mathcal{B}(c, t_c)$ if T has at least one node in $\mathcal{B}(c, t_c \cdot \sigma)$, the weight of T is at most $\phi \cdot L$, and sum of the profits of the clients got hit by T is at least $\frac{1}{\omega}$OPT where OPT is the amount of profit an optimum tree collects with no violation in any bound.*

Accepting a tri-criteria approximation algorithm for NPCST has two benefits. Firstly, it reduces the difficulty of solving the NPCST problem to avoid the hardness results similar to Theorem 2. Secondly, later when we transform the solution of the algorithm to a solution of SUM-MR, it allows the approximation factor on the traveling time for a client and a repairman to reach to a certain node (latency) to get split between both the client (violating its neighborhood) and the repairman (violating the weight of the tree). However a solution to NPCST resulting from a tri-criteria approximation algorithm is harder to transform to a solution of SUM-MR. We prove the following general theorem to transform any tri-criteria approximation algorithm to the NPCST problem to an efficient approximation algorithm for SUM-MR.

Theorem 3. *Given a* (σ, ϕ, ω)*-approximation algorithm for NPCST, we can find an* $O(\max(\sigma, 2\phi) \cdot \omega)$*-approximation algorithm for* SUM-MR*.*

Proving Theorem 3 has two parts. The first part is to use the tri-criteria approximation algorithm to find a feasible solution for our LP and the second part is to round the feasible solution. For the first part we introduce a new relaxed LP for SUM-MR to absorb the violations of the tri-criteria approximation algorithm while keeping the optimal value of the relaxed LP to be at most the optimal value of the original LP. Then we show that using the (σ, ϕ, ω)-approximation algorithm for NPCST, we can find a feasible solution to the relaxed LP with the objective value at most the optimum value of the original LP.

For the second part of the proof, we round the feasible solution found in the previous part to an integral solution for SUM-MR with the total latency at most $O(\max(\sigma, 2\phi) \cdot \omega)$ times the optimal value of our LP. Our algorithm (later given in Figure 1) for the rounding part is easy to state and implement but needs a relatively complicated analysis. The algorithm runs in several steps where each step represents a time-stamp. The time-stamps increase geometrically, *i.e.,* the time-stamp of a step is twice as the time-stamp of the previous step. At each step we randomly select a tour for each repairman from the set of all tours with the length at most the time-stamp times the repairman's speed. The random selection is done using the LP values. The output of our algorithm is the concatenation of all the tours selected at each step. The idea for the analysis of our algorithm is as follows. Let the time-stamp for a certain step be 2^a and F be the (fractional) number of clients that are served by time 2^a according to the LP values. We show the expected number of clients that our algorithm serves in the step is at least $\frac{3F}{4}$. We show that this condition is enough to bound the total latency of the clients. Finally, we show that our algorithm can be derandomized. The derandomization is done by a recursive algorithm which takes an arbitrary subset (R') of R and selects a path for a repairman r in R' and calls itself with parameter $R' \setminus \{r\}$. It selects a path for r which covers the maximum number of clients from the set of clients that are served fractionally by repairmen of R' in the LP solution but not served by the paths we have selected till now. We prove that the greedy algorithm serves at least $\lceil \frac{3F}{4} \rceil$ clients by induction on the size of set R'.

We prove the following theorem about the NPCST problem for the general metrics which is of independent interest and non-trivial. In order to prove Theorem 1, we plug this result about the NPCST problem in Theorem 3.

Theorem 4. *There is an* $(O(\log n), O(\log n), 2)$*-approximation algorithm for the NPCST problem in general metrics.*

Remember that in the NPCST problem we have to find a tree T to maximize the number of clients whose $\mathcal{B}$-ball contain a node of T. To prove the above theorem we embed the graph into a distribution of tree metrics[FRT04]. Note that the $\mathcal{B}$-balls for the clients are not preserved in the tree metrics. We define a new problem on the tree metrics as follows. Given a budget L' we want to find a tree T with weight at most L to maximize the size of the set of served clients C' where T serves set C' if the sum of distances of the clients in C' to T is at most L'. We solve the new problem efficiently in the tree metrics with dynamic programming. Finally we show by violating the radii of the $\mathcal{B}$-balls of the clients by a constant factor, T actually serves a good fraction of the clients in C' (by Markov's Inequality) through hitting their $\mathcal{B}$-balls.

An anonymous referee pointed out that we can also obtain an $(O(1), O(1), O(\log n))$-approximation algorithm for the NPCST problem in general metrics using the ideas in [GKK$^+$01] and [SK04]. This algorithm is interesting since as opposed to the algorithm of Theorem 4 there is no violation on the cost of the tree and the neighborhoods' radii but it collects an $O(\log n)$ fraction of the optimal profit. Note that by plugging this algorithm into Theorem 3 we get the same result for the NPCST problem as in Theorem 1. The description of the algorithm and an outline of the proof given by the referee is brought in the full version of this paper [HKKK13].

Motivated from the application of Sum-MR in Amazon Locker Delivery [Ama10] and USPS gopost [Ser10] which occurs in the Euclidean space, we prove the following theorem to get a constant-factor approximation algorithm for Sum-MR in the Euclidean space. The neighborhood problems are also especially studied in the geometric settings. The usual assumption in the neighborhood TSP problems for getting a constant factor approximation is to assume the radius of the biggest neighborhood is at most a constant factor larger than the smallest one. We plug the following theorem in Theorem 3 to get a constant factor approximation algorithm for Sum-MR. Here the radius constraints means the maximum speed of the clients is at most a constant factor larger than the minimum speed which is an acceptable constraint for the package delivery problem motivating Sum-MR.

Theorem 5. *There is an* $(O(P), O(1), O(1))$*-approximation algorithm for the NPCST problem in the Euclidean space where the radius of the greatest neighborhood is at most* P *times larger than the radius of the smallest neighborhood.*

Our last result is a constant-factor approximation algorithm for Max-MR.

Theorem 6. *There is a constant-factor approximation algorithm for the Max-MR problem when the repairmen have the same speed.*

3 The Sum Movement Repairmen Problem

3.1 Preliminaries

First we formalize the conditions that have to be met in order to serve the clients. If a client collocates with a repairman at a node u of the metric space at time t we say it is served *perfectly* with latency t. On the other hand, if a repairman visits a node u at time t and a client visits u at time $t' \geq t$ we say the client is served *indirectly* with latency t'.

The following lemma shows serving indirectly instead of perfectly does not change the total latency by more than a constant factor.

Lemma 1. *Suppose a solution (sol) to* Sum-MR *has sum of latencies l where all the clients are served indirectly, then sol can be transformed to a solution (sol′) in which all the clients are served perfectly with sum of latencies at most $(3+\epsilon)\cdot l$ where $\epsilon > 0$ is a fixed constant.*

Proof. Remember Definition 1, solution *sol* assigns a path p_i to each repairman r_i and a node u_j to each client c_j such that c_j can go to u_j by time t_j while a repairman has visited u_j before or at time t_j. Suppose repairman r_i can travel p_i in t_i units of time considering its speed. In other words, the length of p_i is at most $v_i \cdot t_i$ where v_i is the speed of r_i. When r_i serves the clients indirectly it is better for him to travel p_i as fast as possible and does not wait for the clients to arrive since the clients can arrive in the nodes of p_i later and are still served indirectly.

We design the movements in *sol′* as follows. The movement for the clients are the same as *sol*, each client c_j is assigned to same node u_j as in *sol* and go to the assigned node by time t_j. Each repairman r_i starts from depot s_i (its starting node) and goes one unit of time along path p_i with its maximum speed v_i and comes back to s_i, we refer to this as round 0, then it goes α units of time and comes back to s_i (round 1) where $\alpha = 1 + \frac{2}{\epsilon}$. In general at each round x it travels α^x units of time along p_i and comes back. If at round y the given time α^y is enough to travel p_i completely, repairman r_i travels p_i completely and stays at the last node to finish time α^y and then comes back to s_i[1].

Now we prove that if a client is served indirectly with latency q in *sol* it will be served perfectly with latency $(3+\epsilon)q$ in *sol′*. Suppose an arbitrary client c_j is served indirectly with r_i at time q in a node u_j of path p_i. Note that q either represents the time when both r_i and c_j arrive at u_j or the time when c_j arrives at u_j but r_i is already passed u_j. In *sol′*, when c_j reaches u_j it waits for repairman r_i to visit u_j after or at time q during its back and forth travels.

Note that each round x takes $2\alpha^x$ units of time. Repairman r_i can serve c_j perfectly the first time it visits u_j after time q. The first $\lfloor \log_\alpha q \rfloor$ rounds take

[1] In fact, if at round y repairman r_i comes back to s_i as soon as it finishes traveling p_i results in a better total latency in some cases. We avoid this because it is harder to analyze and explain. Moreover, in the worst case the total latency remains the same.

$\sum_{x=0}^{\lfloor \log_\alpha q \rfloor} 2\alpha^x$ units of time which is equal to $2(\alpha^{\lfloor \log_\alpha q \rfloor + 1} - 1)/(\alpha - 1)$ and hence greater than q. Therefore r_i serves c_j perfectly at most at round $\lfloor \log_\alpha q \rfloor + 1$. At round $\lfloor \log_\alpha q \rfloor + 1$, repairman r_i needs at most another q units of time to reach to u_j. Therefore when r_i travels at most $\sum_{x=0}^{\lfloor \log_\alpha q \rfloor} 2\alpha^x + q$ units of time, it visits u_j and serves c_j perfectly. Thus, the latency of client c_j getting served perfectly is at most $\sum_{x=0}^{\lfloor \log_\alpha q \rfloor} 2\alpha^x + q \leq \left(3 + \frac{2}{\alpha - 1}\right) q$ which is equal to $(3 + \epsilon) \cdot q$ by replacing back $\alpha = 1 + \frac{2}{\epsilon}$. The lemma follows by applying the same argument to all the clients. □

We focus on finding a solution to the Sum-MR problem where the clients are served indirectly and then transform it to a solution which serves the clients perfectly using Lemma 1. Therefore, from now on whenever we use serving we mean serving indirectly.

We start with some important definitions.

Definition 4. *Let neighborhood $\mathcal{B}(c, t_c)$ denote the set of all nodes whose distances from s'_c, the starting node of client c, are at most t_c.*

Definition 5. *Let $\mathcal{P}(r, t_r)$ denote the set of all non-simple paths (i.e., they can visit nodes or edges multiple times) with length at most t_r starting from s_r, the starting depot of r.*

Using the above two definition we can formalize the notion of serving as follows.

Definition 6. *We call a repairman r serves client c or client c getting served by r at time t if the path selected for r hits neighborhood $\mathcal{B}(c, v'_c \cdot t)$ where v'_c is the speed of client c.*

Let mv be the maximum speed of all the clients and repairmen. We multiply all the edges of the graph by $2 \cdot mv$ which scales all the service times by factor $2 \cdot mv$. Now we can assume that the minimum service time a client can see is at least 1. Let T be the largest service time a client can see, here we upper bound T to be $\frac{2 \cdot MST(G)}{\min_i v_i}$ which is the units of time required to travel all the edges by the slowest repairman and hence serving all the clients. We use set $Q = \{1, 2, \ldots, 2^i, \ldots, 2^{\lceil \log T \rceil + \lceil \log m \rceil / 2 + 1}\}$ to index geometrically increasing time-stamps. The greatest element of Q is chosen such that all the clients are guaranteed to be served by our algorithm after this time-stamp. Note that we have $\lceil \log T \rceil + \lceil \log m \rceil / 2 + 1$ elements in Q and hence its size is polynomially bounded by the size of input.

3.2 LP formulation for Sum-MR

In this section we introduce an LP formulation for the Sum-MR problem and show how to solve this LP approximately. We use the following LP for Sum-MR inspired by the ideas from LPs introduced by Chakrabarty and Swamy [CS11].

$$\min \quad \sum_{c\in C}\sum_{t\in Q} t\cdot y_{c,t} \qquad (\mathbf{PLP})$$

$$s.t. \quad \sum_{p\in\mathcal{P}(r,v_r\cdot t)} x_{r,p,t} \le 1 \qquad \forall t\in Q, \forall r\in R \qquad (1)$$

$$\sum_{r\in R}\sum_{p\in\mathcal{P}(r,v_r\cdot t):p\cap\mathcal{B}(c,v'_c\cdot t)\neq\emptyset} x_{r,p,t} \ge \sum_{t'\le t} y_{c,t'} \qquad \forall c\in C, \forall t\in Q \qquad (2)$$

$$\sum_{t\in Q} y_{c,t} \ge 1 \qquad \forall c\in C \qquad (3)$$

$$x, y \ge 0 \qquad (4)$$

The variable $x_{r,p,t}$ is the indicator variable showing whether repairman r travels path $p \in \mathcal{P}(r, v_r \cdot t)$ completely by time t. Note that if p is in set $\mathcal{P}(r, v_r \cdot t)$, from the definition of $\mathcal{P}(r, v_r \cdot t)$, r can complete traveling p within time t. Variable $y_{c,t}$ is the indicator variable showing if client c is served at time t.

Constraints (3) guarantee that every client gets served. Constraints (1) require each repairman r to travel at most one path by time t. The amount $\sum_{t'\le t} y_{c,t'}$ shows the fraction of service, client c demands until time t and the amount $\sum_{r\in R}\sum_{p\in\mathcal{P}(r,v_r\cdot t):p\cap\mathcal{B}(c,v'_c\cdot t)\neq\emptyset} x_{r,p,t}$ shows the fraction of service c gets from the repairmen until time t. Constraints (2) guarantee that the total service from all the repairmen is at least as large as the demand from the client c. Note that Constraints (2) just require the total demand by client c should be served by the repairmen at anytime before time t which is the case for serving indirectly.

Note that we only consider times that are in the set Q but in a solution of Sum-MR the clients may be served at any time.

Lemma 2. *The optimal value of* **PLP** *is at most twice the optimal solution of* Sum-MR.

Proof. Before proving the lemma, note that we assumed that the smallest element of Q (the smallest latency seen by the clients) is one. Remember it does not change our analysis since the time-stamps in Q grow exponentially which guarantees that the size of Q is polynomial in terms of the inputs.

Intuitively the factor two comes from the fact that we only consider the powers of 2 for the time-stamps in set Q, because if a client must served at time t in an optimal solution in the LP it might wait for the next power of two to get served which can be at most $2t$. More formally, we show every integral solution ($\hat{sol}$) to the Sum-MR problem with total latency ℓ^* can be transformed to a feasible solution $(\hat{x}, \hat{y})$ to **PLP** with the objective value at most $2\ell^*$. If repairman r_i travels path p_i by time t_i in $\hat{sol}$ we set $\hat{x}_{r_i,p_i,t_i} = 1$. If client c is served at time t in $\hat{sol}$, we set $y_{c,t'} = 1$ where $t' = 2^{\lceil \log t\rceil}$. We set all the other non-set entries of $(\hat{x}, \hat{y})$ to zero. The objective value for $(\hat{x}, \hat{y})$ is at most twice the total latency of $\hat{sol}$ because if a client (c) is served at time t in $\hat{sol}$, c contributes at most t' in the objective value for $(\hat{x}, \hat{y})$ where $t' < 2t$ since $2^{\lceil \log t\rceil} < 2^{\log t+1}$. Thus the optimal value of **PLP** is at most twice the total latency of an optimal solution for Sum-MR. □

In the next subsection we show that we are able to find a solution to the SUM-MR problem which has total latency at most $O(\log n)$ times the optimum value of **PLP** which along with Lemma 2 upper bounds the integrality gap of the LP, where $n = |V|$ is the number of nodes in the metric space.

3.3 Solving PLP in Polynomial Time

The first difficulty to solve **PLP** is that it has exponentially many variables. In order to solve the LP we formulate its dual. The dual LP has exponentially many constraints but polynomially many variables therefore we need a separation oracle for the constraints in order to solve the dual LP in polynomial time. The dual LP for **PLP** is as follows.

$$\max \quad \sum_{c \in C} \lambda_c - \sum_{r \in R, t \in Q} \beta_{r,t} \qquad \textbf{(DLP)}$$

$$s.t. \quad \sum_{c: p \cap \mathcal{B}(c, v'_c \cdot t) \neq \emptyset} \theta_{c,t} \leq \beta_{r,t} \qquad \forall r \in R, \forall t \in Q, \forall p \in \mathcal{P}(r, v_r \cdot t) \quad (5)$$

$$\lambda_c \leq t + \sum_{t \leq t'} \theta_{c,t'} \qquad \forall c \in C, \forall t \in Q \quad (6)$$

$$\lambda, \beta, \theta \geq 0 \quad (7)$$

We have exponentially many Constraints (5), therefore we need a separation oracle for them in order to use Ellipsoid algorithm to solve **DLP**. Given a candidate solution (λ, β, θ) for any repairman $r_i \in R$ and time-stamp $t \in Q$ we define Separation Oracle Problem $SOP(r_i, t)$ as follows. Assume that each client c has profit $\theta_{c,t}$ and $\mathcal{B}$-ball $\mathcal{B}(c, v'_c \cdot t)$. The objective is to find a path in $\mathcal{P}(r_i, v_i \cdot t)$ (has maximum length $t \cdot v_i$) which collects the maximum profit where a path collects the profit of any client whose $\mathcal{B}$-ball is hit by the path. If for all $r_i \in R$ and $t \in Q$ the optimal path collects at most $\beta_{r,t}$ profits, there is no violating constraint and (λ, β, θ) is a feasible solution; otherwise there exists a separating hyperplane.

The separation oracle explained above is NP-Hard since it contains the orienteering problem as a special case where the radius of all the $\mathcal{B}$-balls are zero. Therefore, we can only hope for an approximate solution for the separation oracle unless $P = NP$.

Note that $SOP(r_i, t)$ is the same as instance $(V, d, r_i, C, t \cdot v_i)$ of the NPCST problem (Definition 2) except instead of finding an optimum tree we have to find an optimum path. Because paths are the special cases of the trees, the optimum value for the NPCST instance is at least the optimum value of $SOP(r_i, t)$. Therefore, if we solve the NPCST instance we collect at least the same amount of profit. We will use the $\big(O(\log n), O(\log n), 2\big)$-approximation algorithm in Theorem 4 to solve the NPCST instance and transform the resulting tree to a path by doubling the edges and taking an Eulerian tour which increases the length of the path by a factor of 2. In fact, we approximately solve $SOP(r_i, t)$ by violating the budget on the resulting path, the radius of clients' $\mathcal{B}$-balls, and not collecting the maximum profit.

Due to all the violations explained above on the constraints of $SOP(r_i, t)$ we cannot bound the objective value of the feasible solution resulting from the $\big(O(\log n), O(\log n), 2\big)$-approximation algorithm. To this end, we introduce a relaxation of **PLP** ($PLP^{(\mu,\omega)}$) in the following, when μ, ω are constant integers greater than or equal to 1.

$$\min \quad \sum_{c\in C}\sum_{t\in Q} t \cdot y_{c,t} \qquad (PLP^{(\mu,\omega)})$$

$$s.t. \quad \sum_{p\in\mathcal{P}(r,\mu\cdot v_r\cdot t)} x_{r,p,t} \le \omega \qquad \forall r \in R, \forall t \in Q \qquad (8)$$

$$\sum_{r\in R}\sum_{p\in\mathcal{P}(r,\mu\cdot v_r\cdot t):p\cap\mathcal{B}(c,\mu\cdot v'_c\cdot t)\neq\emptyset} x_{r,p,t} \ge \sum_{t'\le t} y_{c,t'} \qquad \forall c \in C, \forall t \in Q \qquad (9)$$

$$\sum_{t\in Q} y_{c,t} \ge 1 \qquad \forall c \in C \qquad (10)$$

$$x, y \ge 0 \qquad (11)$$

Constraint (8) is the same as Constraint (1) except instead of $\mathcal{P}(r, v_r \cdot t)$ we have $\mathcal{P}(r, \mu \cdot v_r \cdot t)$ which allows repairman r to take a path which is μ times longer than a regular path in $\mathcal{P}(r, v_r \cdot t)$. Moreover by putting ω instead of 1 we allow each repairman to take ω routes instead of one. Constraint (9) is the same as Constraint (2) except instead of $\mathcal{B}(c, v'_c \cdot t)$ we have $\mathcal{B}(c, \mu \cdot v'_c \cdot t)$ and instead of $\mathcal{P}(r, v_r \cdot t)$ we have $\mathcal{P}(r, \mu \cdot v_r \cdot t)$ which allow both the repairmen and clients to take paths that are μ times longer.

In the following lemma we show a (σ, ϕ, ω)-approximation algorithm for NPCST can be used to find a feasible solution to $PLP^{(\mu,\omega)}$ whose cost is at most the optimum solution of **PLP**. The proof of this lemma which is provided in detail in the full version of the paper [HKKK13] is relatively involved and is quite more general than a lemma used in [CS11].

Lemma 3. *Given a (σ, ϕ, ω)-approximation algorithm for NPCST, one can find a feasible solution to $PLP^{(\mu,\omega)}$ in polynomial time, where $\mu = \max(\sigma, 2 \cdot \phi)$, with objective value at most* $\text{OPT}(1 + \epsilon)$ *for any $\epsilon > 0$ where* OPT *is the optimal value of* **PLP**.

3.4 Rounding the LP

We show how to use feasible solution (x, y) taken from Lemma 3 to obtain an integral solution to Sum-MR with the total latency at most $O(\max(\sigma, 2\phi) \cdot \omega \cdot \text{OPT})$ and thus finish the proof of Theorem 3. Sum-Movement Repairmen Algorithm (Sum-MRA) shown in Figure 1 is our algorithm to do so.

1. For each $q \in Q$ in ascending order do:
   (a) For each $f \in [4 \cdot \omega]$ do:
      i. Perform Append-path(q, f).

Append-path(q, f):

1. For each repairman r, select a path ($p_r^f \in \mathcal{P}(r, \mu \cdot v_r \cdot q)$) independently at random where the probability of path p gets selected is equal to value $\frac{1}{\omega} x_{r,p,q}$ computed from $PLP^{(\mu,\omega)}$. Each repairman r travels its selected path and comes back to its starting depot.
2. Serve all the clients, that are yet unserved but can reach to a node in a selected path by time $q \cdot \mu$.

Fig. 1. Movement Repairmen Algorithm (Sum-MRA)

As explained earlier Q in Sum-MRA is the set $\{1, 2, \ldots, 2^i, \ldots, 2^{\lceil \log T \rceil + \lceil \log m \rceil / 2 + 1}\}$ where T is the latest service time a client can see which was upper bounded by $\frac{2 \cdot MST(G)}{\min_i v_i}$ and the value $\lceil \log T \rceil + \lceil \log m \rceil / 2 + 1$ is chosen to guarantee that Sum-MRA serves all the clients after it finishes. Moreover, μ and ω are the constants in $PLP^{(\mu,\omega)}$.

Sum-MRA serves clients in multiple steps. It starts serving clients with paths that have the maximum latency $1 \cdot \mu$ then it concatenate paths of maximum latency $2 \cdot \mu$, then $4 \cdot \mu$ and so on. These paths come from the set $\mathcal{P}(r, \mu \cdot v_r \cdot q)$ for $q \in Q$ and the selection is done using $PLP^{(\mu,\omega)}$ variables $x_{r,p,q}$. In fact, for each $q \in Q$ we select $4 \cdot \omega$ paths by executing Append-path(q, f) $4 \cdot \omega$ times where f is just used to iterate over set $[4 \cdot \omega]$. This is because we want to have independence between the paths selected at each execution of Append-path(q, f) which helps us to better analyze the number of clients get served in the execution.

We use the following definitions to refer to the clients served by Sum-MRA.

Definition 7. *Let $A^{q,f}$ denotes the set of non-served clients getting served at Instruction 2 of Append-path(q, f). i.e., $A^{q,f}$ is the set of clients c such that c is not served before the execution of Append-path(q, f) but it can reach a node v by time $\mu \cdot q$ such that there exists a repairman r with $v \in p_r^f$ (remember p_r^f is the path selected for r in Append-path(q, f)).*

Definition 8. *Set $\mathcal{A}^{q,f} = \bigcup_{(q',f') \leq (q,f)} A^{q',f'}$ is the set of all clients served by* Sum-MRA *up to and including the execution of Append-path(q, f). Here the operator $\leq$ is the lexicographic ordering for the ordered pairs where the first entry has more priority than the second one.*

We define function $prev(q, f)$ as follows.

$$prev(q, f) = \begin{cases} (q, f-1) & f \neq 1 \\ (\frac{q}{2}, 4 \cdot \omega) & f = 1 \end{cases}$$

Definition 9. *Let $(q', f') = prev(q, f)$ and Append-path(q', f') be the predecessor of Append-path(q, f). Let $F^{q,f}$ denote the value of $\sum_{c \in C \setminus \mathcal{A}^{q',f'}} \sum_{t \le q} y_{c,t}$. Intuitively, $F^{q,f}$ can be taught as the fractional number of clients that are (fractionally) served in feasible solution (x, y) by the time q, but not served by* Sum-MRA *before the execution of Append-path(q, f).*

We would like in $A^{q,f}$, be a large fraction of $F^{q,f}$. First we prove the following lemma to lower bound the probability of a client getting served in the execution of Append-path(q, f).

Lemma 4. *Let q be any element of Q and c be any client in C. If we randomly select a path for each repairman $r \in R$ such that the probability of selecting $p \in \mathcal{P}(r, \mu \cdot v_r \cdot q)$ is $\frac{1}{\omega} \cdot x_{r,p,q}$, then the probability of c getting served (a selected path visits a node from $\mathcal{B}(c, \mu q)$) is at least $\frac{1}{2\omega} \cdot \sum_{q' \le q} y_{c,q'}$.*

Proof. The probability of a client c getting served by an arbitrary repairman $r \in R$ is $D_r = \frac{1}{\omega} \sum_{p \in \mathcal{P}(r, \mu \cdot v_r \cdot t) : p \cap \mathcal{B}(c, \mu \cdot v'_c \cdot t) \neq \emptyset} x_{r,p,t}$ from the probability distribution used in the rounding. To simplify the notations, let $B = \sum_{r \in R'} D_r$ and $Y_c = \frac{1}{\omega} \sum_{q' \le q} y_{c,q'}$.

The probability that a client c is not served by any repairman in R is $\prod_{r \in R}(1 - D_r)$.

$$
\begin{aligned}
\prod_{r \in R}(1 - D_r) &\le \left(\frac{|R| - \sum_{r \in R} D_r}{|R|} \right)^{|R|} && \text{Arithmetic and Geometric Means Inequality } [2] \\
&= \left(1 - \frac{B}{|R|} \right)^{|R|} && \text{(replacing by } B\text{)} \\
&= \left(1 - \frac{1}{\frac{|R|}{B}} \right)^{\frac{|R|}{B} B} \\
&\le e^{-B} \\
&\le e^{-Y_c} && \text{Constraint (9)}
\end{aligned}
$$

From the above inequality we conclude that client c gets served with probability at least $1 - e^{-Y_c}$. The following inequalities finish the proof of the lemma.

[2] For any set of n non-negative numbers $x_1, \ldots, x_n$ we have $\frac{x_1 + \ldots + x_2}{n} \ge \sqrt[n]{x_1 \cdot x_2 \cdot \ldots \cdot x_n}$

$$
\begin{aligned}
1 - e^{-Y_c} &= 1 - \left(\sum_{i=0}^{\infty} \frac{(-Y_c)^i}{i!} \right) && \text{by Taylor Expansion} \\
&\geq Y_c - \frac{Y_c^2}{2} && \text{as } 0 \leq Y_c \leq 1 \\
&\geq \frac{1}{2} Y_c && \text{as } 0 \leq Y_c \leq 1 \\
&\geq \frac{1}{2\omega} \cdot \sum_{q' \leq q} y_{c,q'} && \text{definition of } Y_c
\end{aligned}
$$

□

We use the following lemma (proof appears in the full version of the paper [HKKK13]) to derandomize selections of the paths in Append-path(q, f) and to show that $A^{q,f}$ is at least $\left\lceil \frac{F^{q,f}}{2 \cdot \omega} \right\rceil$.

Lemma 5. *We can derandomize Append-path(q, f) to deterministically select a path $p_r^f \in \mathcal{P}(r, q \cdot v_r \cdot \mu)$ for each repairman r, such that the set of newly served clients ($A^{q,f}$ as defined in Definition 7) to be at least $\left\lceil \frac{F^{q,f}}{2 \cdot \omega} \right\rceil$.*

We prove the following lemma in the full version of the paper [HKKK13] which combined with Lemma 3 finishes the proof of Theorem 3.

Lemma 6. *A feasible solution (x, y) to the $PLP^{(\mu,\omega)}$ with objective value* OPT *(the optimum value for* **PLP***) can be rounded to an integral solution to* SUM-MR *with total latency $O(\mu \cdot \omega) \cdot$ OPT.*

Acknowledgment. The authors would like to thank an anonymous reviewer of Approx+Random 2013 conference who provided us with helpful comments and more importantly an alternative proof which gives an $(O(1), O(1), O(\log n))$-approximation algorithm for the NPCST problem.

References

[Ama10] Amazon. Amazon locker delivery (2010), `http://www.amazon.com/gp/help/customer/display.html?nodeId=200689010`

[BCC+94] Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., Sudan, M.: The minimum latency problem. In: ACM Symposium on Theory of Computing, pp. 163–171 (1994)

[BYCR93] Baeza-Yates, R.A., Culberson, J.C., Rawlins, G.J.E.: Searching in the plane. Information and Computation 106, 234 (1993)

[CGRT03] Chaudhuri, K., Godfrey, B., Rao, S., Talwar, K.: Paths, trees, and minimum latency tours. In: IEEE Symposium on Foundations of Computer Science, pp. 36–45 (2003)

[CK04] Chekuri, C., Kumar, A.: Maximum coverage problem with group budget constraints and applications. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) APPROX and RANDOM 2004. LNCS, vol. 3122, pp. 72–83. Springer, Heidelberg (2004)

[CS11] Chakrabarty, D., Swamy, C.: Facility location with client latencies: Linear programming based techniques for minimum latency problems. In: Günlük, O., Woeginger, G.J. (eds.) IPCO 2011. LNCS, vol. 6655, pp. 92–103. Springer, Heidelberg (2011)

[FHR03] Fakcharoenphol, J., Harrelson, C., Rao, S.: The k-traveling repairman problem. In: ACM-SIAM Symposium on Discrete Algorithms, pp. 655–664 (2003)

[FRT04] Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. Journal of Computer and System Sciences 69(3), 485–497 (2004)

[GK98] Goemans, M., Kleinberg, J.: An improved approximation ratio for the minimum latency problem. Mathematical Programming 82(1), 111–124 (1998)

[GKK+01] Gupta, A., Kleinberg, J., Kumar, A., Rastogi, R., Yener, B.: Provisioning a virtual private network: a network design problem for multicommodity flow. In: Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing, pp. 389–398 (2001)

[HKKK13] Hajiaghayi, M.T., Khandekar, R., Reza Khani, M., Kortsarz, G.: Approximation algorithms for movement repairmen. arXiv:1306.3739 [cs.DS] (2013)

[Ser10] The United States Postal Service. gopost (2010), `https://tools.usps.com/go/EPLAction!input`

[Sit06] Sitters, R.: The minimum latency problem is NP-hard for weighted trees. In: Cook, W.J., Schulz, A.S. (eds.) IPCO 2002. LNCS, vol. 2337, pp. 230–239. Springer, Heidelberg (2002)

[SK04] Swamy, C., Kumar, A.: Primal–dual algorithms for connected facility location problems. Algorithmica 40(4), 245–269 (2004)

[WAW93] Will, T.G., Adviser-West, D.B.: Extremal results and algorithms for degree sequences of graphs (1993)

Improved Hardness of Approximating Chromatic Number

Sangxia Huang

KTH Royal Institute of Technology
Stockholm, Sweden
sangxia@csc.kth.se

Abstract. We prove that for sufficiently large K, it is NP-hard to color K-colorable graphs with less than $2^{\Omega(K^{1/3})}$ colors. This improves the previous result of K versus $K^{\frac{1}{25}\log K}$ in Khot [1].

1 Introduction

A vertex coloring of a graph $G(V, E)$ is an assignment of colors to its vertices such that no two adjacent vertices receive the same color. The minimum number of colors needed for such a coloring is called the chromatic number of G, denoted by $\chi(G)$. As a classical combinatorial optimization problem, vertex coloring is closely related to other problems such as finding maximum independent sets and probabilistically checkable proofs (PCPs) with certain special properties. In addition to being an important theoretical challenge, graph coloring also has a number of applications such as scheduling and register allocation.

It is known that determining the chromatic number of a graph exactly is NP-hard [2]. However, in many applications, it suffices to find a good enough approximation. In other words, given a K-colorable graph, we would like to color it with as few colors as possible. Wigderson [3] gave an algorithm using $O(n^{1-1/(K-1)})$ colors. This was improved by Berger and Rompel [4] to $O\left((n/\log n)^{1-1/(K-1)}\right)$ colors. Karger, Motwani and Sudan [5] used semi-definite programming to achieve $\tilde{O}(n^{1-3/(K+1)})$, which was adapted in Blum and Karger [6] to an algorithm that colors a 3-colorable graph with $\tilde{O}(n^{3/14})$ colors. For 3-colorable graphs, the best known algorithm is by Kawarabayashi and Thorup [7] which uses $O(n^{0.2038})$ colors, based on results by Arora and Chlamtac [8] and Chlamtac [9].

There have been many works on the hardness side as well. It is known that coloring 3-colorable graphs with 4 colors is NP-hard, and for general K-colorable graphs it is NP-hard to color with $K + 2\lfloor \frac{K}{3} \rfloor - 1$ colors [10,11]. For sufficiently large K, the best known gap is by Khot [1] which proved that it is NP-hard to color a K-colorable graph with $K^{\frac{1}{25}\log K}$ colors. Assuming a variant of Khot's 2-to-1 Conjecture, Dinur, Mossel and Regev [12] proved that it is NP-hard to K'-color a K-colorable graph for any $3 \leq K < K'$. The dependency between the hardness of graph coloring and the parameters of 2-to-1 Label Covers was

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 233–243, 2013.

made explicit and improved by Dinur and Shinkar [13], who showed that it is NP-hard to $\log^c n$-color a 4-colorable graph for some constant $c > 0$ assuming the 2-to-1 Conjecture. Guruswami and Sinop [14] proved that assuming the 2-to-1 Conjecture, it is NP-hard to find an independent set with more than $O\left(\frac{n}{\Delta^{1-c/(k-1)}}\right)$ vertices in a k-colorable graph of maximum degree Δ for some absolute constant $c \leq 4$.

Khot's hardness result [1] can be derived either using PCPs from Håstad and Khot [15] or Samorodnitsky and Trevisan [16]. We can view the results in both works as showing approximation resistance for a family of Boolean predicates that have very few accepting inputs — it is NP-hard to approximate Max CSP better than just picking random assignments on instances whose constraints are expressed with those predicates. For each integer $k > 0$, the approximation resistant predicates we get from [15] and [16] have k variables (and thus 2^k possible assignments) but only have $2^{O(\sqrt{k})}$ accepting assignments. The predicate from Håstad and Khot [15] is approximation resistant even on satisfiable instances — or have perfect completeness in PCP language — while the predicate from Samorodnitsky and Trevisan [16] is not. It is noted in Khot [1] that having perfect completeness is not necessary but makes the reduction for coloring easier.

In a recent breakthrough, Chan [17,18] proved approximation resistance for a family of predicates on k variables but only has $k + 1$ accepting assignments whenever k is of the form $k = 2^r - 1$. Previously, approximation resistance of those predicates was only known assuming the Unique Games Conjecture, proved by Samorodnitsky and Trevisan [19]. Hast [20] proved that predicates on k variables having at most $2\lfloor k/2 \rfloor + 1$ ($= k$ in the current setting) accepting inputs are not approximation resistant, thus these results are almost tight.

In [17], Chan also showed that for any $K \geq 3$, there is $\nu = o(1)$ such that given a graph with an induced K-colorable subgraph of fractional size $1 - \nu$, it is NP-hard to find an independent set of fractional size $1/2^{K/2} + \nu$. Although this gives a larger gap than Khot [1], the result lacks "perfect completeness" and thus is not comparable with Khot [1]. We refer to [21,22,17] for additional discussions on Almost-Coloring.

In this paper, we show improved hardness of approximating chromatic number using the above results.

Theorem 1. *For all sufficiently large K, it is NP-hard to color a K-colorable graph with $2^{\Omega(K^{1/3})}$ colors. Moreover, this hardness result holds for graphs that have degree bounded by $O(K2^{K^{1/3}})$.*

Stated in terms of degree, Theorem 1 says that there exists some constant c, such that for all large enough Δ, it is NP-hard to color a $(\log \Delta)^3$-colorable graph of maximum degree bounded by Δ with $O(\Delta^c)$ colors.

Our approach follows that of Khot [1]. The main part of the work is to adapt Khot's technique so that it works with the new PCP construction by Chan [17], which gives much better dependency between soundness and the arity of the Max CSP, and is also the main source of the improvement in Theorem 1. This reduction alone will give us graphs with degree at least doubly exponential in

K. To get a tighter dependency of degree, we apply a technique in Trevisan [23] to sparsify the output of the reduction.

2 Preliminaries

In this section we review the basics of Label Cover and PCPs and describe Chan's improved PCP construction.

Let (U, V, E, L, R, Π) be an instance of Label Cover, where $R = dL$ for some constant d, the tuple (U, V, E) is a bipartite graph, vertices in U are assigned labels from $[L]$, and vertices in V are assigned labels from $[R]$. Each edge $e = (u, v)$ is associated with a d-to-1 mapping $\pi_e : [R] \to [L]$. Given a labeling $A : U \to [L], V \to [R]$, we say that the constraint on e is satisfied if $\pi_e(A(v)) = A(u)$. The value of a labeling is the fraction of edges that are satisfied, and the value of a Label Cover instance is the maximum value over all possible labelings of its vertices. The following theorem combines the celebrated PCP theorem [24,25] with Raz's parallel repetition theorem [26] and shows hardness of Label Cover.

Theorem 2. *For any constant $0 < \sigma < 1$, there are $d, L \leq \text{poly}(1/\sigma)$ such that the problem of deciding satisfiability of a 3-SAT instance with n variables can be Karp-reduced in* $\text{poly}(n)$ *time to the problem of deciding whether a Label Cover instance of size* $\text{poly}(n)$ *has value 1 or at most σ. The graph in Label Cover is a bi-regular bipartite graph with left- and right-degrees* $\text{poly}(1/\sigma)$.

As is the case with many inapproximability results, the above Label Cover will be the starting point of our reduction. Formally, let $P : \{-1, 1\}^k \to \{-1, 1\}$ be a Boolean predicate of arity k, where we follow the convention of having -1 as "True" and 1 as "False". In a Max-P problem, we are given an instance on n Boolean variables $x_1, \cdots, x_n$ with m clauses. All clauses have form $P(l_1, \cdots, l_k)$, where each literal l_i is either a variable or its negation, and the variable of the literals are distinct. The goal of the Max-P problem is to find an assignment to $x_1, \cdots, x_n$ that maximizes the number of clauses satisfied by the assignment. The reduction from Label Cover to Max-P typically translates labelings for $u \in U$ and $v \in V$ to $2^{|L|}$ and $2^{|R|}$ Boolean variables, respectively. These variables are viewed as functions $f^u : \{-1, 1\}^{|L|} \to \{-1, 1\}$ and $g^v : \{-1, 1\}^{|R|} \to \{-1, 1\}$. We require that these functions are folded, that is, for any $x \in \{-1, 1\}^{|L|}$, $y \in \{-1, 1\}^{|R|}$, $f^u(-x) = -f^u(x)$ and $g^v(-y) = -g^v(y)$. For each pair of queries $(x, -x)$, we select one of them. If x is selected, then when $f(-x)$ is needed we return $-f(x)$ instead. Hence in the actual reduction we only use $2^{|L|-1}$ Boolean variables for each $u \in U$ and $2^{|R|-1}$ variables for each $v \in V$. This is also why we need to allow negated literals in the CSP instances. In a correct proof for a satisfiable Label-Cover instance, the functions are long codes for the corresponding labelings of u and v, that is, having $f^u(x) = x_{\sigma_U(u)}$, and $g^v(y) = y_{\sigma_V(v)}$.

Now we describe the clauses in Max-P. For an edge (u, v) in the Label-Cover, we sample *queries*

$$(x^{(1)}, \cdots, x^{(m)}, y^{(m+1)}, \cdots, y^{(k)})$$

according to some carefully chosen *test distribution* $\mathcal{T}$. The distribution $\mathcal{T}$ has the property that for any $l \in L$ and $r \in R$ such that $\pi_{(u,v)}(r) = l$, the predicate P accepts $(f^u(x_l^{(1)}), \cdots, f^u(x_l^{(m)}), g^v(y_r^{(m+1)}), \cdots, g^v(y_r^{(m+1)}))$ with probability 1 (or $1-\varepsilon$ for some constant ε if we are considering non-perfect completeness). One can verify that if the Label Cover instance has value 1 and the test distribution $\mathcal{T}$ satisfies the above property, then any correct proof of a correct labeling has the required completeness. In the soundness analysis, we are given functions f^u and g^v that achieve non-trivial acceptance probability in the above test, and we need to decode those functions and obtain non-trivial labelings of the original Label Cover instance.

In [17], Chan developed a new way of constructing efficient PCPs and proved that the following Hadamard predicate $H_K : \{-1,1\}^K \to \{-1,1\}$ is approximation resistant. For $K = 2^r - 1$, the predicate H_K is on variables $\{x_S\}_{\emptyset \neq S \subseteq [r]}$, defined as

$$H_K(x) = \begin{cases} -1 & \forall S \subseteq [r], |S| > 1, x_S = \prod_{i \in S} x_{\{i\}} \\ 1 & \text{otherwise.} \end{cases}$$

This predicate has $K+1$ accepting assignments. Samorodnitsky and Trevisan [19] showed that H_K is approximation resistance assuming the Unique Games Conjecture — a conjecture stating that finding an approximately optimal solution for a certain special kind of Label Cover is NP-hard. Using his new technique, Chan proved that this is true assuming only $P \neq NP$.

The main idea in Chan's reduction is to consider a direct sum of PCPs. We now sample K edges and run K independent copies of the above test. In the i-th PCP, the i-th query is a uniform random string from $\{-1,1\}^{|L|}$ and all other queries are sampled from $\{-1,1\}^{|R|}$ as described below in Definition 2. In a correct proof, the strategies are expected to be products of long codes encoding the labeling of the vertices.

We now formally define the PCP and how queries are sampled. In the following description, we identify integers from $[K]$ and non-empty subsets of $[r]$ in some canonical way. First we describe the test distribution for a single PCP, indexed by non-empty sets $\emptyset \neq S \subseteq [r]$.

Definition 1. *Let e_S be an edge and π be the constraint on e. Denote the set of possible queries to the T-th position by Q_T, where*

$$Q_T = \begin{cases} \{-1,1\}^{|L|} & T = S \\ \{-1,1\}^{|R|} & T \neq S. \end{cases}$$

The test distribution $\mathcal{T}_{S,e_S}$ is a distribution on $\prod_{T \subseteq [r]} Q_T$. To sample query $(q_T)_{T \subseteq [r]}$ from $\mathcal{T}_{S,e_S}$, first sample q_S from $\{-1,1\}^{|L|}$ uniformly at random. Then, for each $i \in [R]$, let $\{q_{T,i}\}_{T \neq S}$ be a uniformly random accepting assignment of H_K, conditioned on the S-th bit being equal to $q_{S,\pi(i)}$. Finally, independently for each bit, we add noise by resampling from the uniform distribution on $\{-1,1\}$ with probability η.

The final test distribution in the PCP is a product of the above distribution.

Definition 2. *Let (U, V, E, L, R, Π) be a label cover instance. For $i \in [K]$, define $\mathcal{V}_i = V^{i-1} \times U \times V^{K-i}$. For each $\mathbf{v} \in \mathcal{V}_i$, the proof contains function $\mathbf{f}_\mathbf{v} : \left(\{-1,1\}^R\right)^{i-1} \times \{-1,1\}^L \times \left(\{-1,1\}^R\right)^{K-i} \to \{-1,1\}$. The verifier checks the proof as follows:*

1. *Sample independently $K = 2^r - 1$ uniformly random edges $\{e_S\}_{\emptyset \neq S \subseteq [r]}$. Denote $e_S = (u_S, v_S)$.*
2. *Sample queries $\{\mathbf{q}_i\}_{i=1}^K$ from distribution $\prod_{\emptyset \neq T \subseteq [r]} \mathcal{T}_{T,e_T}$.*
3. *Let $\mathbf{v}_i = (v_1, \cdots, v_{i-1}, u_i, v_{i+1}, \cdots, v_K)$. Accept if*

$$H_K(\mathbf{f}_{\mathbf{v}_1}(\mathbf{q}_1), \cdots, \mathbf{f}_{\mathbf{v}_K}(\mathbf{q}_K)) = -1 \ .$$

In a correct proof, the function $\mathbf{f}_\mathbf{v}$ is the product of long codes encoding the labeling of each vertex in $\mathbf{v}$.

Remark. As in the ordinary case, we require that the functions $\mathbf{f}_\mathbf{v}$ are folded in the following sense — for any $j \in [K]$, query $\{\mathbf{q}_{j,i}\}_{i \in [K]}$ and $i_0 \in [K]$ we have

$$\begin{aligned} & \mathbf{f}_\mathbf{v}(\mathbf{q}_{j,1}, \cdots, -\mathbf{q}_{j,i_0}, \cdots, \mathbf{q}_{j,K}) \\ = \ & -\mathbf{f}_\mathbf{v}(\mathbf{q}_{j,1}, \cdots, \mathbf{q}_{j,i_0}, \cdots, \mathbf{q}_{j,K}). \end{aligned}$$

Theorem E.1 along with Theorem A.1, 6.9 and C.2 of Chan [18] shows completeness and soundness of the above reduction and we summarize in the following theorem.

Theorem 3. *Fix some small $\eta, \delta > 0$. Let σ be the soundness of Label Cover, satisfying $\delta = poly(K/\eta) \cdot \sigma^{\Omega(1)}$. Given a Label Cover instance $LC_{L,dL}$, we have the following:*

1. *If $LC_{L,dL}$ has value 1, the above verifier accepts a correct proof with probability at least $1 - K^2\eta$.*
2. *If $LC_{L,dL}$ has value at most σ, then given any proof the verifier accepts with probability at most $(K+1)/2^K + 2\delta$.*

Let $\varepsilon > 0$ be some small constant. In the rest of the paper, let $\delta = \varepsilon \cdot 2^{-K}$, and $\eta = \varepsilon/K^2$. By Theorem 3, we require the soundness of Label Cover to be $\sigma = (\delta/\text{poly}(K/\eta))^{O(1)} = 2^{-\Omega(K)}$. This means that the size of the label $L = \text{poly}(1/\sigma) = \exp(\Theta(K))$.

3 Hardness of Approximating Chromatic Number

In this section, we prove Theorem 1 — for sufficiently large K, it is NP-hard to color a K-colorable graph with less than $2^{\Omega(K^{1/3})}$ colors. For convenience of notation, we in fact prove a gap of $O(K^3)$ versus $2^{\Omega(K)}$.

The overall approach follows that in Khot [1]. We start by describing the FGLSS graph [27] of the PCP as described in Definition 2. The vertices in the FGLSS graph are function queries and corresponding accepting configurations, denoted as $(\mathbf{f}_\mathbf{v}, \mathbf{q}, \mathbf{z})$. The weight of the vertex is the probability that query

$(\mathbf{f_v}, \mathbf{q})$ is picked. The total weight of the graph is therefore $K+1$, the number of accepting assignments of H_K. Two vertices are connected if they are clearly inconsistent (returning different answers for the same query to the same function). An independent set in the graph corresponds to a strategy / set of functions, and its weight is the acceptance probability of such strategy. Note that if the maximum weight independent set of the FGLSS graph has weight w, then we need at least $(K+1)/w$ colors to color the whole graph since vertices having the same color must form an independent set.

To use the FGLSS graph for coloring results, we also need to show that if a PCP has acceptance probability $1-\varepsilon$, we can color the FGLSS graph with a small number of colors. Note that in this case, we know that there is an independent set of weight $1-\varepsilon$ in the FGLSS graph, corresponding to a correct proof. Khot's idea in [1] is to modify the definition of the PCP so that the correct proofs are parameterized by some global parameter $\alpha \in \{0,1\}^t$. This gives us 2^t different correct proofs and thus 2^t independent sets of weight $1-\varepsilon$, and by choosing the right t, we expect those independent sets cover most of the FGLSS graph of the modified PCP and thus gives a coloring of at most 2^t colors.

Formally, we modify Definition 2 so that the functions in the proof become $\mathbf{f}_{\mathbf{v}_i} : \left(\{-1,1\}^{R\cdot 2^t}\right)^{i-1} \times \{-1,1\}^{L\cdot 2^t} \times \left(\{-1,1\}^{R\cdot 2^t}\right)^{K-i} \to \{-1,1\}$. Alternatively, we can think of this as modifying Label Cover by appending a t-bit binary string to all the labels and defining the new projection in the Label Cover instance as $\pi'_e(r \circ \alpha) = \pi_e(r) \circ \alpha$ for $r \in R$ and $\alpha \in \{0,1\}^t$, where "$\circ$" denotes string concatenation. The value of this new Label Cover instance is exactly the same as the original setting. Consider the FGLSS graph in this new setting. Soundness is straightforward. If the new proof makes the verifier accept with probability at least $(K+1)/2^K + 2\delta$, then by Theorem 3, the value of the new Label Cover is at least σ and hence the original instance also has value at least σ.

Now let us consider the case of completeness. If the original Label Cover instance has value 1, then extending a valid labeling with any $\alpha \in \{0,1\}^t$ gives us a valid labeling for the modified instance, which corresponds to an independent set of weight at least $1-\varepsilon$ in the modified FGLSS graph. We need to show that the 2^t independent sets corresponding to different $\alpha \in \{0,1\}^t$ cover almost all of the FGLSS graph of the modified PCP. In fact, we can efficiently identify a small fraction of the vertices that contains all vertices that are not covered by any independent sets of the above form and remove them from the FGLSS graph.

To this end, we follow Khot's notation and introduce the following definition characterizing whether we can cover certain vertex with independent sets.

Definition 3. *Consider any K tuples of labelings $\mathbf{l} = \{(l_i, r_i)\}_{i=1}^K$, where $l_i \in [L]$, $r_i \in [R]$ for all $i \in [K]$. Define the i-th mixed labeling*

$$\mathbf{m}_i(\mathbf{l}) = (r_1, \cdots, r_{i-1}, l_i, r_{i+1}, \cdots, r_K) \ .$$

Let $\mathbf{f}_{i,\mathbf{l}}$ be the product of long codes encoding the labelings in $\mathbf{m}_i$. Denote by $\mathbf{l}^\alpha := \{(l_i \circ \alpha, r_i \circ \alpha)\}_{i=1}^K$ the labelings extended by α. Define $\mathbf{f}^\alpha_{i,\mathbf{l}}$ similarly.

A set of queries $\mathbf{q} = (q_1, \cdots, q_K)$ *is good if for any* K *tuples of labelings* $\mathbf{l}$ *and any accepting assignment* $\mathbf{z} = (z_1, \cdots, z_K)$ *of the Hadamard predicate, there exists a global extension* α*, such that* $\mathbf{f}^{\alpha}_{i,\mathbf{l}}(q_i) = z_i$ *for all* $i \in [K]$.

To verify if a set of queries is good, we only need to check all K tuples of labelings and all accepting assignments of the Hadamard predicate H_K. Those are all constants depending only on K (and ε). The following lemma shows that the fraction of bad queries is small.

Lemma 1. *Let* t *be such that* $2^t = C \cdot K^3$ *for some large constant* C*. For large enough* K*, at most a weighted fraction of* $\exp(-\Theta(K))$ *of the queries is not good.*

Before proving the lemma, let us see how it leads to our main theorem.

Remove the vertices in the FGLSS graph that correspond to queries that are not good. By Lemma 1, the fraction of vertices removed is bounded by $\exp(-O(K))$. In the soundness case coloring the FGLSS graph still needs at least $(K+1)(1-\exp(-\Theta(K)))/2^{-K} = 2^{\Omega(K)}$ colors. In the completeness case, the Label Cover instance has value 1. Fix a labeling that satisfies all the edges. For a vertex $(\mathbf{f_v}, \mathbf{q}, \mathbf{x})$ in the modified FGLSS graph, let $\mathbf{l_v}$ be the set of K tuples of labelings of the sampled vertices. Each $\alpha \in \{0,1\}^t$ is associated with an independent set consisting of vertices of the form $(\mathbf{f_v}, \mathbf{q}, \mathbf{z})$, where $z_i = \mathbf{f}^{\alpha}_{i,\mathbf{l_v}}(q_i)$ for all $i \in [K]$.

Consider any vertex $(\mathbf{f_v}, \mathbf{q}, \mathbf{x})$ in the modified FGLSS graph. We know that $\mathbf{q}$ is good so by definition there exists $\alpha_0 \in \{0,1\}^t$ such that $\mathbf{f}^{\alpha_0}_{i,\mathbf{l_v}}(q_i) = x_i$ for all $i \in [K]$. Hence, it is covered by the independent set associated with α_0. Therefore the modified FGLSS graph can be colored with $2^t = O(K^3)$ colors.

Proof (Proof of Lemma 1.). For query $\mathbf{q}$, let $Q(\mathbf{q})$ be the event that $\mathbf{q}$ is not good in the sense of Definition 3: there exists some labeling $\mathbf{l}$ and some accepting assignment $\mathbf{z}$, such that for any α, there exists $i \in [K]$, $\mathbf{f}^{\alpha}_{i,\mathbf{l}}(q_i) \neq z_i$. It suffices to bound $\Pr_{\mathbf{q}}[Q(\mathbf{q})]$.

Fix some K tuples of labeling $\mathbf{l}$ of the label cover instance and some accepting assignment $\mathbf{z}$. Consider $\alpha \in \{0,1\}^t$. Over the queries sampled, the probability that $\mathbf{f}^{\alpha}_{i,\mathbf{l}}(q_i) = z_i$ for all $i \in [K]$ is $1/(K+1)$ before adding noise. To estimate the effect of noise, note that there are K functions, each being a product of K long codes, therefore the answers $\{\mathbf{f}^{\alpha}_{i,\mathbf{l}}(q_i)\}_{i\in[K]}$ depends on K^2 bits. If none of these K^2 bits are corrupted, then the answer is exactly $\mathbf{z}$. This gives an overall probability of $\Theta(1/K \cdot (1-\eta)^{K^2}) = \Theta(e^{-\eta K^2}/K) = \Theta(1/K)$. The contribution of probability from other sources is negligible.

Note that for different extension α, the bits that $\mathbf{f}^{\alpha}_{i,\mathbf{l}}$ reads from $\mathbf{q}$ are completely independent, so we have

$$\Pr_{\mathbf{q}}\left[\forall \alpha, \exists i, \mathbf{f}^{\alpha}_{i,\mathbf{l}}(q_i) \neq z_i\right] = (1-\Theta(1/K))^{2^t} = \exp(-\Theta(2^t/K)) \ .$$

Picking large enough constant C and taking union bound over all possible labelings and accepting configurations, we get that the weighted fraction of $\mathbf{q}$ that are bad is

$$\Pr_{\mathbf{q}}[Q(\mathbf{q})] \leq R^{K-1} \cdot L \cdot (K+1)\exp(-\Theta(2^t/K)) = \exp(-\Theta(K)) \ .$$

Now let us consider the degree of the graph produced by the above reduction. Consider a vertex $(\mathbf{f_v}, \mathbf{q}, \mathbf{z})$. Fix some $i \in [K]$. Let $\mathbf{z}'$ be some accepting assignment of H_K with $z'_i \neq z_i$. We first estimate the number of queries $\mathbf{q}'$ with $q'_i = q_i$. Let us consider the i-th test distribution $\mathcal{T}_{i,e_i}$, where e_i is the edge sampled for the i-th test, and denote the constraint on e_i by π. Recall that for each $l \in [L]$ and $r \in \pi^{-1}(l) \subseteq [R]$, the bits $\{q'_{j,r}\}_{j \neq i}$ are sampled by uniformly picking an accepting assignment $\mathbf{x}$ of H_K conditioned on $x_i = q'_{i,l}$. Thus there are at least $((K+1)/2)^{|R|} = 2^{\exp(\Omega(K))}$ possible choices of $\mathbf{q}'$. Note that for any such $\mathbf{q}'$, there is an edge between $(\mathbf{f_v}, \mathbf{q}', \mathbf{z}')$ and $(\mathbf{f_v}, \mathbf{q}, \mathbf{z})$. Therefore the degree of the graph produced by the above reduction is at least double exponential in K. We now use the approach in Clementi and Trevisan [28] and Trevisan [23] to reduce the degree to $O(K^3 2^K)$.

For ease of presentation, we look at the argument on the original FGLSS graph without removing bad queries. The same argument applies to the graph with bad queries removed because removing vertices from the graph does not increase the maximum degree, and, as seen above, does not significantly affect the soundness and completeness of the reduction.

Denote the FGLSS graph corresponding to the PCP in Theorem 3 as G. We first turn G into an unweighted graph. Let w_{min} be the minimum weight of vertices in G, and λ be the ratio between the minimum and maximum weight of vertices in G. Since in the test distribution in Definition 2 edges of the Label Cover instance are sampled uniformly, we have that λ depends only on K. Let ξ be some granularity parameter. We obtain an unweighted version G' of G by duplicating vertices — we make $\lfloor w/w_{min} \cdot 1/\xi \rfloor \leq 1/\lambda\xi$ vertices for a vertex of weight w, and connect the duplicated vertices with all the neighbors. This step blows up the size of the graph by $O(1/\lambda^2\xi^2)$, and the fractional size of the maximum independent set in G' is within a multiplicative factor of $O(\xi)$ from that of G due to error introduced by $\lfloor \cdot \rfloor$ when duplicating vertices.

As observed in [23], the graph G' is a union of bipartite complete subgraphs. More precisely, for every index i and i-th query $(\mathbf{f}_{\mathbf{v}_i}, \mathbf{q}_i)$, there is a complete bipartite graph between configurations that answer zero for query $(\mathbf{f}_{\mathbf{v}_i}, \mathbf{q}_i)$ — denoted as $Z_{\mathbf{f}_{\mathbf{v}_i}, \mathbf{q}_i}$ — and configurations that answer one for the same query — denoted as $O_{\mathbf{f}_{\mathbf{v}_i}, \mathbf{q}_i}$. By the way we construct the FGLSS graph, it follows that these complete bipartite subgraphs cover the whole G'. Let l be the maximum size of such sets. We claim that l depends only on K, λ and ξ. To estimate l, consider how many tuples $(\mathbf{f_v}, \mathbf{q}, \mathbf{z})$ can include $(\mathbf{f}_{\mathbf{v}_i}, \mathbf{q}_i)$ on the i-th position. By Theorem 2, the degree of the Label Cover graph is $\text{poly}(1/\sigma) = \exp(\Theta(K))$, thus the $\mathbf{f}_{\mathbf{v}_i}$ coordinate has at most $\exp(\Theta(K^2))$ neighbors. For $\mathbf{q}_i$, consider an edge e the bits in $\mathbf{q}_i$ that are mapped to the same label $l \in [L]$ according to mapping π_e (or a single bit if e is the i-th edge). There are exactly $(K+1)/2$ possible queries. Enumerating over all labels and sampled edges, this gives an upper-bound of $2^{\exp(\Theta(K))}$. Since each of them can be duplicated by at most $1/\lambda\xi$ times, we have $l = 2^{\exp(\Theta(K))}/\lambda\xi$. Also since for each input bit to the predicate H_K, exactly half of the accepting assignments of H_K set that bit to 1

and exactly half to -1 — a property also known as H_K being balanced — we have $|Z_{\mathbf{f}_{\mathbf{v}_i},\mathbf{q}_i}| = |O_{\mathbf{f}_{\mathbf{v}_i},\mathbf{q}_i}|$.

We now replace the above bipartite complete graphs in G' with the following construction on the same set of vertices $Z_{\mathbf{f}_{\mathbf{f}_i},\mathbf{q}_i}$ and $O_{\mathbf{f}_{\mathbf{f}_i},\mathbf{q}_i}$.

Proposition 1 ([23]). *For every $\zeta > 0$ and $b \geq 1$, there is a bipartite graph $([b], [b], E)$ of degree at most $d = 3\zeta^{-1}\log(\zeta^{-1})$ such that for any $A, B \subseteq [b]$, $|A| \geq \lfloor \zeta b \rfloor$, $|B| \geq \lfloor \zeta b \rfloor$, we have $(A \times B) \cap E \neq \emptyset$.*

Trevisan [23] called such graphs (b, ζ)-dispersers, and he used a probabilistic argument to prove the above proposition. As argued above, l is a constant depending only on K, thus we can find the desired disperser by exhaustive search. An important property of bipartite dispersers is that given an independent set I of a (b, ζ)-disperser, we have that either $|I \cap A| \leq \zeta b$ or $|I \cap B| \leq \zeta b$.

Denote the replaced graph by G''. To understand how much the above replacement step increases the size of the maximum independent set, note that for any independent set in a disperser, we can get an independent set in the complete bipartite graph by discarding all vertices on one side, which is at most a ζ fraction if we choose the smaller side. Also, each vertex in the FGLSS graph is involved in at most K complete bipartite graphs of this kind, thus the size of the independent set in the new graph is at most $K\zeta$ larger than G'. By choosing $\zeta = O(2^{-K}/K)$, $\xi = O(2^{-K})$, we have that in the soundness case the maximum independent set G' has size $O(2^{-K})$. The maximum degree of G'' is bounded by $K \cdot 3\zeta^{-1}\log(\zeta^{-1}) = O(K^3 2^K)$.

4 Discussions

In this paper, we proved a gap of K^3 vs. $2^{\Omega(K)}$ for approximating chromatic number. Let us take a closer look at how we get to the power 3 in K^3. The soundness of the Label Cover problem has to be at most $2^{-\Omega(K)}$, which means that the size of the labels are $\exp(\Theta(K))$. Definition 3 involves all possible labelings and accepting assignments of K. The reduction in Definition 2 samples K edges, therefore there would be $R^{K-1} \cdot L = \exp(\Theta(K^2))$ possible labelings and a union bound results in a factor of $\exp(\Theta(K^2))$ in the probability of a query being bad. The other factor of K is due to the fact that the Hadamard predicate H_K has $K + 1$ accepting assignments and they are sampled uniformly.

Acknowledgments. The author is grateful to Siu On Chan for pointing out Theorem E.1 in [17], leading to a gap of K vs. $2^{\Omega(K^{1/3})}$, improving from a gap of K vs. $2^{\Omega(K^{1/5})}$ in an earlier version of this manuscript.

The author is supported by ERC Advanced Investigator grant 226203.

References

1. Khot, S.: Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In: FOCS, pp. 600–609 (2001)
2. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
3. Wigderson, A.: A new approximate graph coloring algorithm. In: STOC, pp. 325–329 (1982)
4. Berger, B., Rompel, J.: A better performance guarantee for approximate graph coloring. Algorithmica 5(3), 459–466 (1990)
5. Karger, D.R., Motwani, R., Sudan, M.: Approximate graph coloring by semidefinite programming. J. ACM 45(2), 246–265 (1998)
6. Blum, A., Karger, D.R.: An $\tilde{O}(n^{3/14})$-coloring algorithm for 3-colorable graphs. Inf. Process. Lett. 61(1), 49–53 (1997)
7. Kawarabayashi, K., Thorup, M.: Combinatorial coloring of 3-colorable graphs. In: FOCS, pp. 68–75 (2012)
8. Arora, S., Chlamtac, E.: New approximation guarantee for chromatic number. In: Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing, STOC 2006, pp. 215–224. ACM, New York (2006)
9. Chlamtac, E.: Approximation algorithms using hierarchies of semidefinite programming relaxations. In: FOCS, pp. 691–701 (2007)
10. Khanna, S., Linial, N., Safra, S.: On the hardness of approximating the chromatic number. Combinatorica 20(3), 393–415 (2000)
11. Guruswami, V., Khanna, S.: On the hardness of 4-coloring a 3-colorable graph. SIAM J. Discrete Math. 18(1), 30–40 (2004)
12. Dinur, I., Mossel, E., Regev, O.: Conditional hardness for approximate coloring. SIAM J. Comput. 39(3), 843–873 (2009)
13. Dinur, I., Shinkar, I.: On the conditional hardness of coloring a 4-colorable graph with super-constant number of colors. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, pp. 138–151. Springer, Heidelberg (2010)
14. Guruswami, V., Sinop, A.K.: The complexity of finding independent sets in bounded degree (hyper)graphs of low chromatic number. In: SODA, pp. 1615–1626 (2011)
15. Håstad, J., Khot, S.: Query efficient PCPs with perfect completeness. Theory of Computing 1(1), 119–148 (2005)
16. Samorodnitsky, A., Trevisan, L.: A PCP characterization of NP with optimal amortized query complexity. In: STOC, pp. 191–199 (2000)
17. Chan, S.O.: Approximation resistance from pairwise independent subgroups. In: STOC, pp. 447–456 (2013)
18. Chan, S.O.: Approximation resistance from pairwise independent subgroups. Electronic Colloquium on Computational Complexity (ECCC) 19, 110 (2012)
19. Samorodnitsky, A., Trevisan, L.: Gowers uniformity, influence of variables, and PCPs. SIAM J. Comput. 39(1), 323–360 (2009)
20. Hast, G.: Beating a random assignment. PhD Thesis (2005)
21. Dinur, I., Khot, S., Perkins, W., Safra, M.: Hardness of finding independent sets in almost 3-colorable graphs. In: FOCS, pp. 212–221 (2010)
22. Khot, S., Saket, R.: Hardness of finding independent sets in almost q-colorable graphs. In: FOCS, pp. 380–389 (2012)

23. Trevisan, L.: Non-approximability results for optimization problems on bounded degree instances. In: STOC, pp. 453–461 (2001)
24. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM 45(3), 501–555 (1998)
25. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. J. ACM 45(1), 70–122 (1998)
26. Raz, R.: A parallel repetition theorem. SIAM J. Comput. 27(3), 763–803 (1998)
27. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. J. ACM 43(2), 268–292 (1996)
28. Clementi, A.E.F., Trevisan, L.: Improved non-approximability results for minimum vertex cover with density constraints. Theor. Comput. Sci. 225(1-2), 113–128 (1999)

A Pseudo-approximation for the Genus of Hamiltonian Graphs

Yury Makarychev[1,⋆], Amir Nayyeri[2,⋆⋆], and Anastasios Sidiropoulos[3,⋆⋆⋆]

[1] Toyota Technological Institute at Chicago
[2] Carnegie Mellon University
[3] University of Illinois at Urbana–Champaign

Abstract. The genus of a graph is a very basic parameter in topological graph theory, that has been the subject of extensive study. Perhaps surprisingly, despite its importance, the problem of approximating the genus of a graph is very poorly understood. It has been shown to be NP-complete by Thomassen [Tho89], and the best known upper bound for general graphs is an $O(n)$-approximation that follows by Euler's characteristic.

We give a polynomial-time pseudo-approximation algorithm for the orientable genus of Hamiltonian graphs. More specifically, on input a graph G of orientable genus g, and a Hamiltonian path in G, our algorithm computes a drawing into a surface of either orientable, or non-orientable genus $g^{O(1)}$.

1 Introduction

A *drawing* of a graph G into a surface $\mathcal{S}$ is a mapping that sends every vertex $v \in V(G)$ into a point $\varphi(v) \in \mathcal{S}$, and every edge into a simple curve connecting its endpoints, so that the images of different edges are allowed to intersect only on their endpoints. A surface is called *orientable* if it can be embedded into $\mathbb{R}^3$, and *non-orientable* otherwise. The *genus* of a graph G is the minimum $g \geq 0$, such that G can be drawn into a surface of genus g. Similarly, the *orientable* (resp. *non-orientable*) genus of a graph is the minimum genus of an orientable (resp. non-orientable) surface into which G can be drawn.

Drawings of graphs into various surfaces are of central importance in graph theory (e.g. [GT01, MT01]), topology, and mathematics in general (e.g. [Whi01]), and have been the subject of intensive study. Graphs of small genus are also of great importance in computer science, and engineering, since they can be used to model a wide variety of natural objects. For further background, we refer the reader to Gross and Tucker [GT01] for topological graph theory and to Hatcher [Hat02] for algebraic topology.

⋆ Supported in part by the NSF Career Award CCF-1150062.
⋆⋆ Supported by NSF under Grants No. CCF 1065106 and CCF 09-15519. Portions of this work were done while the author was a student at the Univ. of Illinois at Urbana Champaign and while he was visiting Toyota Tech. Inst. at Chicago.
⋆⋆⋆ Supported in part by David and Lucille Packard Fellowship, NSF AF award CCF-0915984, and NSF grant CCF-0915519.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 244–259, 2013.

Computing the genus of a graph exactly. It was shown by Thomassen that computing the orientable genus [Tho89], and the non-orientable genus of a graph [Tho93], are both NP-complete problems. Deciding whether a graph has genus 0, i.e. planarity testing, can be done in linear time by the seminal result of Hopcroft & Tarjan [HT74]. Filotti et al. [FMR79] were the first to obtain an algorithm for computing a drawing of an n-vertex graph of genus g, into a minimum-genus surface, in time $n^{O(g)}$. For fixed g, Robertson & Seymour [RS90], as part of their Graph Minor project, gave an $O(n^3)$ time algorithm for determining the genus of a graph. This was improved by a breakthrough result of Mohar [Moh99] who gave a linear-time algorithm for computing a minimum-genus drawing, for any fixed g. A relatively simpler linear-time algorithm has been subsequently obtained by Kawarabayashi, Mohar & Reed [KiMR08]. The running time of the above algorithms is at least exponential in g.

Approximating the genus of a graph. Perhaps surprisingly, the problem of approximating the genus of a graph is very poorly understood. In general, the genus of a graph can be as large as $\Omega(n^2)$ (e.g. for the complete graph K_n). Euler's characteristic implies that any n-vertex graph of genus g has at most $O(n+g)$ edges. Since any graph can be drawn into a surface that has one handle for every edge, this immediately implies a $O(n/g)$-approximation, which is a $\Theta(n)$-approximation in the worst case. In other words, even though we currently cannot exclude the existence of a $O(1)$-approximation, the state of the art only gives a trivial $O(n)$-approximation. We also remark that by Euler's formula, there is a trivial $O(1)$-approximation for sufficiently dense graphs (i.e. of average degree at least $6+\varepsilon$, for some fixed $\varepsilon > 0$).

For graphs of bounded degree, better results are known. Chen, Kanchi, and Kanevsky [CKK97] described a simple $O(\sqrt{n})$-approximation for graphs of bounded degree, which follows by the fact that graphs of small genus have small balanced vertex-separators. Following the present paper, Chekuri & Sidiropoulos [CS] obtained a polynomial time algorithm which given a graph G of bounded degree, and of genus g, outputs a drawing into a surface of genus $g^{O(1)} \log^{O(1)} n$. Combined with the result of Chen et al., this implies a $n^{1/2-\alpha}$ approximation for bounded-degree graphs, for some constant $\alpha > 0$.

What about graphs of unbounded degree? The *only* positive result prior to our work for a non-trivial family of graphs of *unbounded degree*, is due to Mohar [Moh01], who gave an elegant characterization of the genus of apex graphs (i.e. graphs that can be made planar by the removal of a single vertex). This can in turn be used to obtain a $O(1)$-approximation for such graphs. Perhaps surprisingly, except for the above result for apex graphs, essentially *all* currently known approximation algorithms for genus [CKK97, CS], and related parameters, such as crossing number [LR99, CMS11, Chu11, CH11, PH10, MC08], and edge-planarization [LR99, Chu11, CS], apply only to the case of graphs of bounded degree. In many cases, the reason for this phenomenon appears to be that an algorithm relies heavily on divide & conquer via balanced edge-separators. However, graphs of small genus, and unbounded degree, are not guaranteed to have

small edge-separators. This seemingly small technical difference, is a major obstacle towards obtaining approximation algorithms for general graphs. To the best of our knowledge, our result is the most general approximation algorithm for *any* of the above graphs parameters, that works on graphs of unbounded degree.

Our results. We present a pseudo-approximation algorithm for the orientable genus of Hamiltonian graphs.[1] More specifically, we obtain a polynomial-time algorithm which given a graph G, and a Hamiltonian path P in G, computes a drawing of G into a surface of either orientable, or non-orientable genus $O(g^7)$, where g is the orientable genus of G. The dependence of the running time is polynomial in both g, and n. Combined with the simple $O(n/g)$-approximation described above, our result immediately gives a $O(n^{6/7})$-pseudo-approximation for the orientable genus of Hamiltonian graphs. The following summarizes our main result.

Theorem 1.1. *There exists a polynomial-time algorithm which given a graph G of orientable genus g, and a Hamiltonian path P in G, outputs a drawing of G into a surface of either orientable, or non-orientable genus $O(g^7)$. The running time is polynomial in both n, and g.*

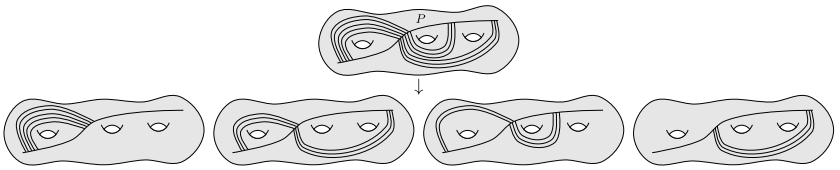
1.1 Overview

In this section, we present a very informal and somewhat imprecise overview of our algorithm. We are given a Hamiltonian graph G of genus g and a Hamiltonian path P in G. Our high-level approach is to *cover* the graph G by $O(g)$ subgraphs of constant genus, then independently draw each of them on a surface of small genus and finally combine all drawings. More precisely, our algorithm consists of the following steps:

Step 1: Cover G by $O(g)$ toroidal subgraphs $G_1, \dots, G_k$.
Step 2: For each pair of graphs G_i, G_j, compute a drawing of $G_i \cup G_j$.
Step 3: Combine the drawing of all pairs $G_i \cup G_j$, to obtain a drawing of G.

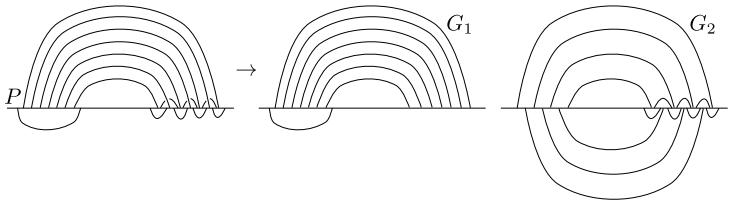
Step 1: Greedy band covering. The goal in this step is to cover G be $O(g)$ toroidal graphs. Fix an optimal drawing of G into a surface of genus g. Let us sat that an edge $e \in E(G) \setminus E(P)$ is *global* if after contracting P, e becomes a noncontractible loop. Otherwise, we say that e is *local*. We can show that G can be covered by a collection of toroidal graphs $G_1^*, \dots G_k^*$, $k = O(g)$, such that every local edge appears in exactly one G_i^*, and every global edge appears in exactly two graphs G_i^*, G_j^*. Roughly speaking, we walk along the path, and we find maximal edge-disjoint subpaths $Q_1, \dots, Q_k$ in P, such that all edges on either side of each Q_i are homotopic (after contracting P). Below is an example of such a covering. For clarity, we omit local edges from the figure.

[1] We use the term *pseudo-approximation* to denote the fact that even though we approximate the orientable genus of the input graph, the output surface is allowed to be non-orientable.



We refer to the graphs $G_1, \ldots, G_k$ as *elementary bands*. A key property is that we can compute such a covering by elementary bands $G_1, \ldots, G_k$, for some $k = O(g)$, even if we do not have access to a drawing of G. Of course, this means that the elementary bands $G_1, \ldots, G_k$ we compute might differ from $G_1^*, \ldots, G_k^*$. This introduces certain complications as we explain next.

Step 2: Drawing a pair of bands. Even though we have $O(g)$ graphs G_i and each of them has genus at most 1, we cannot naively combine their drawings. Roughly speaking, the problem is that graphs G_i share global edges: the way an edge e is drawn in the drawing of G_i may be inconsistent with the way it is drawn in the drawing of G_j. This can happen because in Step 1, when we greedily compute the covering by elementary bands, we can only keep track of the local edges in the current band. As a result, when we try to combine the drawings of two bands, the local edges can introduce conflicts between the two drawings. The following figure depicts an example of such a conflict:



In this example, the graph G is covered by two toroidal (in fact, planar) graphs G_1, G_2. However, in the drawing of G_1, all global edges are drawn on one side of P, while in the drawing of G_2 the global edges alternate between the two sides of P. We overcome this obstacle by showing that, roughly speaking, for every pair of bands G_i, G_j, there are drawings that are *nearly consistent*. This is a very technical part of the paper, and we refer the reader to the next sections for a precise definition. We just note here that, at the high level, we decompose each band into $g^{O(1)}$ subgraphs, such that each such subgraph has a certain structure allowing us to find a drawing into a surface of genus $g^{O(1)}$ in polynomial time.

Step 3: Combining the drawings of all pairs. In this last step we combine the drawing of all pairs of bands into a drawing of G. The key idea is that for every pair $G_i \cup G_j$, we can modify its drawing, such that the global edges that are shared between either G_i, or G_j, and some other elementary band G_ℓ, are contained in a small number of homotopy classes (after contracting P). Intuitively, this means that we can combine the drawings of all pairs by introducing a small number of "interface" handles between them. A precise definition appears in the rest of the paper.

1.2 Why a Pseudo-approximation?

Our current algorithm is a pseudo-approximation, which means that given a graph G of orientable genus g, it outputs a drawing into a surface of either orientable, or non-orientable genus $g^{O(1)}$. However, we believe that our approach can be generalized to obtain a (true) approximation algorithm. That is, given a graph of genus g, compute a drawing into a surface of genus $g^{O(1)}$, with the same orientability type as G. Doing so, requires an extension of the definition of an *elementary band*, to account for graphs that can be drawn into the projective plane (i.e. elementary bands of non-orientable genus at most 1). This small modification causes the number of different types of elementary bands to grow by a constant factor. Unfortunately, as a consequence, the (already lengthy) case analysis in the proof becomes dauntingly long. The rest of our proof remains essentially unchanged. We are not aware of a way to simplify this case analysis, so in the interest of clarity we have decided to omit it from the present paper.

1.3 Organization

In Section 3, we show how to find the elementary band covering. In Section 4, we explain how to combine two drawings, by decomposing then into smaller subgraphs. The drawing of these subgraphs is described in the full version of this paper. In Section 5, we put all pieces of our algorithm together, and show how to find a drawing of G. Due to lack of space, we omit many proofs in the conference version of this paper. They will appear in the full version of the paper.

2 Preliminaries

Definition 2.1 (Combinatorial restriction). *Let G_1 be a graph, and let G_2 be a subgraph of G_1. Let f_1 be a drawing of G_1 into some surface $\mathcal{S}_1$. The* combinatorial restriction *of f_1 on G_2 is defined to be the combinatorial drawing f_2 of G_2 induced by defining for every $v \in V(G_2)$, the ordering of the edges incident to v in G_2 to be their in f_1. By gluing a disk along every facial walk in f_2 we obtain a surface $\mathcal{S}_2$. Then this does not cause confusion, we will naturally identify f_2 with the induced drawing of G_2 into $\mathcal{S}_2$. Note that the genus of $\mathcal{S}_2$ can be smaller than the genus of $\mathcal{S}_1$.*

Lemma 2.1 (Malnič & Mohar [MM92]). *Let $\mathcal{M}$ be an orientable surface of genus g, and let $x \in \mathcal{M}$. Let $\mathcal{X}$ be a collection of noncontractible, pairwise nonhomotopic curves, such that for every $C, C' \in \mathcal{X}$, we have $C \cap C' = x$. Then, $|\mathcal{X}| \leq 6g - 3$.*

3 Band Coverings of Hamiltonian Graphs

In this section, we describe a greedy algorithm that partitions the input graph to a set of $O(g)$ *simple* toroidal graphs, which we call *bands*. One of the tools

that we have developed for this section is based on the notion of ribbons and petals. Intuitively, ribbons and petals describe minimal topological subspaces that contain the edges of a band. Due to space limitations, the formal definitions and lemmas of ribbons and petals are described in the appendix.

Definition 3.1 (Bands in Hamiltonian graphs). *Let G be a graph, and let P be a Hamiltonian path in G. Let $B \subseteq E(G) \setminus E(P)$. Let Q be a subpath of P, such that every edge $e \in B$ has at least one endpoint in $V(Q)$. Then, B is called a* band. *We also say that B has* spine P, *and* primary segment Q. *An edge in B is called* global *if exactly one of its endpoints is in $V(Q)$; it is called* local *if both of its endpoints are in $V(Q)$. Note that every set $B \subseteq E(G) \setminus E(P)$ is a band with spine P, and primary segment P.*

Definition 3.2 (Elementary bands). *Let G be a graph, and let P be a Hamiltonian path in G. Let $B \subseteq E(G) \setminus E(P)$ be a band with spine P, and primary segment $Q \subseteq P$. We define the following types of bands.*

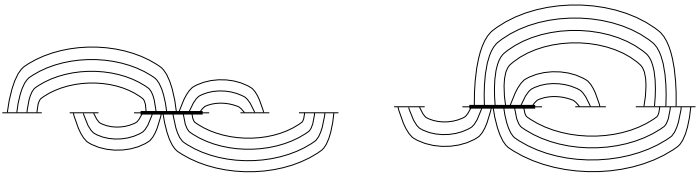
(i) We say that B is of type-1 *(see Figure 1(a)) if the following conditions are satisfied. There exist subpaths $P_1, P_2, P_3, P_4 \subset P$, with P_1, P_2, P_3, P_4, Q being pairwise edge-disjoint, and such that the set M of global edges in B can be decomposed into $M = M_1 \cup M_2 \cup M_3 \cup M_4$, such that for every $i \in \{1, \ldots, 4\}$, every edge in M_i has one endpoint in $V(Q)$, and one in $V(P_i)$. Moreover, there exists a planar drawing φ of the graph $H = B \cup Q \cup P_1 \cup \ldots \cup P_4$, satisfying the following. For every $i \in \{1, \ldots, 4\}$, $\varphi(P_i)$ lies in the outer face, and all the curves $\varphi(e)$, $e \in M_i$, are attached to the same side of $\varphi(P_i)$. We say that the paths $P_1, \ldots, P_4$ are the* outlets *of B.*

(ii) We say that B is of type-2 *(see Figure 1(b)) if the following conditions are satisfied. There exist subpaths $P_1, P_2, P_3 \subset P$, with P_1, P_2, P_3, Q being pairwise edge-disjoint, and such that the set M of global edges in B can be decomposed into $M = M_1 \cup M_2 \cup M_3$, such that for every $i \in \{1, 2, 3\}$, every edge in M_i has one endpoint in $V(Q)$, and one in $V(P_i)$. Moreover, there exists a planar drawing φ of the graph $H = B \cup Q \cup P_1 \cup P_2 \cup P_3$, satisfying the following. If we denote by φ' the drawing induced by φ on $H \setminus V(P_3)$, then for every $i \in \{1, 2\}$, $\varphi'(P_i)$ lies in the outer face. Also, there exists $v \in V(P_3)$, such that $\varphi(v)$ also lies in the outer face. Moreover, for every $i \in \{1, 2\}$, all the curves $\varphi(e)$, $e \in M_i$, are attached to the same side of $\varphi(P_i)$. Note that the curves $\varphi(e)$, $e \in M_3$ are allowed to be attached to both sides of $\varphi(P_3)$. We say that the paths P_1, and P_2 are the* outlets *of B, and that the path P_3 is the* double outlet *of B.*

We say that φ is a canonical drawing *of B (or a canonical drawing of H, when B is clear from the context). For an elementary band of type-2, we can pick φ so that the curves $\varphi(Q)$, $\varphi(P_1)$, $\varphi(P_2)$, $\varphi(P_3)$ become segments of a horizontal line ℓ, with $\varphi(Q)$ appearing to the left of $\varphi(P_3)$. We call ℓ the* canonical line *of φ. Let $\prec$ be the total ordering of $V(Q) \cup V(P_3)$ induced by a left-to-right traversal of ℓ. We say that $\prec$ is the* canonical ordering *of φ. We extend $\prec$ to B as follows. Let $\{x, y\}, \{x', y'\} \in B$, with $x, x' \in V(Q)$. Then, we define $\{x, y\} \prec \{x', y'\}$ if*

either $x \prec x'$, or $(x = x') \wedge (y \prec y')$. Since G does not contain any parallel edges, it follows that $\prec$ is a total ordering of B.

We remark that a band can be of both type-1, and type-2 (e.g. the trivial band). Figure 1 depicts examples of all different types of elementary bands.



(a) Type-1. (b) Type-2.

Fig. 1. The different types of elementary bands. The primary segment is in bold.

Definition 3.3 (Band coverings for Hamiltonian graphs). *Let G be a graph, and let P be a Hamiltonian path in G. A* band covering with spine P *for G is a collection $\mathcal{B} = \{(B_i, Q_i)\}_{i=1}^{t}$ satisfying the following conditions:*

(1) For every $i \in \{1, \ldots, t\}$, B_i is an elementary band with spine P, and primary segment Q_i.
(2) $\bigcup_{i \in \{1,\ldots,t\}} B_i = E(G) \setminus E(P)$.
(3) For every $i \neq j \in \{1, \ldots, t\}$, we have $V(Q_i) \cap V(Q_j) = \emptyset$.

We remark that every edge in $E(G) \setminus E(P)$ is contained in at least one, and at most two bands in the band covering $\mathcal{B}$. If it is contained in exactly one band, then we say that it is local*, and otherwise we say that it is* global*.*

First we show that a band covering that is composed of $O(g)$ bands exists. The main intuition for this proof is looking and the homotopy classes of the edges in $E(G) \setminus P$ in an optimal embedding. Moreover, we show that we can compute a band covering of size $O(g)$. Note that this band covering that we compute is not necessarily optimal, however, its size is within a constant factor of the optimal band covering size. The main tool that our algorithm uses is a so-called *ribbon-petal covering*. See A for the description of ribbon petal covering and the proof of the following lemmas.

Lemma 3.1 (Existence of a small band covering). *Let G be a graph of orientable genus g, and P a Hamiltonian path in G. Then, there exists a band covering $\mathcal{B} = \{(B_i, Q_i)\}_{i=1}^{t}$, with spine P for G, with $t = O(g)$.*

The main technical result of this section is the following.

Lemma 3.2 (Computing a small band covering). *Let G be a graph of orientable genus g, and P a Hamiltonian path in G. Then, given G and P, we can compute in polynomial time a band covering for G with spine P, of size $O(g)$.*

4 Compatible Pairs of Planar Drawings

In the previous section, we show how to decompose the input graph to a collection of toroidal graphs. We use Mohar's algorithm to find an embedding for all those toroidal graphs in linear time. Nevertheless, the computed embeddings are not necessarily consistent. To fix this issue, we consider all pairs of bands and corresponding toroidal embeddings and change them to make their intersections consistent. We obtain a drawing of each pair of bands on a surface of genus $O(g^3)$. In the next section, we show how to resolve the remaining slight inconsistencies to acquire the final embedding. Many of the proofs of this section are presented in the full version of this paper.

Consider two bands B_1 and B_2 with disjoint primary segments Q_1 and Q_2, respectively. Let $B = B_1 \cap B_2$ (the set of edges going from Q_1 to Q_2). Denote the set of local edges incident on vertices in Q_1 by L_1, and the set of edges incident on vertices in Q_2 by L_2 (see Figure 2.).

The main result of this section is Theorem 4.1.

Theorem 4.1. *There is a polynomial time algorithm that finds a drawing of* $P \cup B \cup L_1 \cup L_2$ *on a surface of genus* $O(g^3)$.

We first prove the following decomposition theorem and then show that it implies Theorem 4.1.

Theorem 4.2. *There is a polynomial algorithm that does the following. It divides segments* Q_1 *and* Q_2 *into consecutive segments* $Q_1^1, \ldots, Q_1^s$ *and* $Q_2^1, \ldots, Q_2^s$, *respectively, where* $p = O(g)$. *These segments do not share any vertices except possibly for endpoints. Also it partitions all edges in* B *into* p *disjoint sets* $T_1, ..., T_s$ *such that all edges in* T_i *go between* Q_1^i *and* Q_2^i. *For each* $i \in \{1, \ldots, s\}$, *the algorithm finds a drawing* ψ_i *of* $Q_1 \cup Q_2 \cup T_i \cup L_1 \cup L_2$ *on a surface of genus* $O(g^2)$. *Additionally, the combinatorial restriction of* ψ_i *to* $Q_1 \cup Q_2 \cup L_1 \cup L_2$ *is planar and canonical.*

We consider canonical drawings φ_1 and φ_2 of B_1 and B_2, respectively. Each of the drawings draws paths Q_1 and Q_2 on a horizontal line ℓ and hence defines a total ordering of vertices in Q_1 and Q_2. Let $\prec_1$ be the ordering defined by φ_1 and $\prec_2$ be the ordering defined by φ_2. By changing the orientation of the line ℓ in one of the drawings, if necessary, we may assume that $\prec_1$ and $\prec_2$ agree on $V(Q_1)$. However, orderings $\prec_1$ and $\prec_2$ may define either the same or opposite order on $V(Q_2)$. In the former case, we say that the band intersection is *regular*; in the latter case, we say that the band intersection is *irregular*. In the Appendix, we show that in the irregular case the intersection of bands B_1 and B_2 can be drawn on a surface of genus 8. Thus Theorems 4.1 and 4.2 follow (with $s = 1$).

Lemma 4.1. *In the irregular case, the intersection of bands* B_1 *and* B_2 *can be drawn on a surface of genus 8.*

In the rest of this section, we will analyze the regular case. Since in the regular case orderings $\prec_1$ and $\prec_2$ agree on $V(Q_1) \cup V(Q_2)$, we will just denote this ordering by $\prec$.

Definition 4.1. *Let us say that two edges $\{x, y\}, \{x', y'\} \in B \cup L_1 \cup L_2$ (with $x \prec y$ and $x' \prec y'$) are* in conflict *if $x \prec x' \prec y \prec y'$ or $x' \prec x \prec y' \prec y$. We consider an auxiliary conflict graph $\mathcal{C}$ on the set of edges $B \cup L_1 \cup L_2$ in which e and e' are connected with an auxiliary edge if e and e' are in conflict. We denote the set of connected components of $\mathcal{C}[B]$ (the subgraph of $\mathcal{C}$ induced by B) by $\mathcal{S}$. (See Figure 2.)*

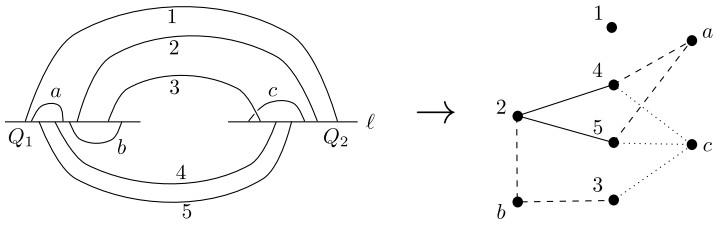


Fig. 2. The figure on the left shows two bands with primary segments Q_1 and Q_2, global edges $B = \{1, 2, 3, 4, 5\}$, local edges $L_1 = \{a, b\}$ and $L_2 = \{c\}$. The right figure shows the corresponding conflict graph $\mathcal{C}$. Note that graphs $\mathcal{C}[B \cup L_1]$ and $\mathcal{C}[B \cup L_2]$ are bipartite as claimed by Lemma 4.2. However, the graph $\mathcal{C}$ is not bipartite since it contains an odd cycle $2 \to 5 \to c \to 3 \to b \to 2$. The graph $\mathcal{C}[B]$ has three connected components $S_1 = \{1\}$, $S_2 = \{2, 4, 5\}$ and $S_3 = \{3\}$ with $S_1 \lhd S_2 \lhd S_3$.

The motivation for Definition 4.1 is that if two edges e and e' are not conflict, we can draw them in the plane on one side of ℓ or on opposite sides of ℓ. However, if two edges are in conflict we can only draw them on one side of ℓ. Accordingly, if the conflict graph is bipartite, then we can simultaneously draw all the edges (the nodes of the conflict graph) in the plane. The following lemma shows that for every $S \in \mathcal{S}$, $\mathcal{C}[S \cup L_1 \cup L_2]$ is bipartite (we prove this lemma in the appendix).

Lemma 4.2. *Graphs $\mathcal{C}[B \cup L_1]$ and $\mathcal{C}[B \cup L_2]$ are bipartite. Moreover, for every $S \in \mathcal{S}$, $\mathcal{C}[S \cup L_1 \cup L_2]$ is bipartite.*

Definition 4.2. *We define a partial order $\lhd$ on edges as follows: $e \lhd e'$ for two edges $e = (x, y)$ and $e' = (x', y')$ if $x \preceq x'$ and $y \succeq y'$ (here, we assume wlog that $x \prec y$ and $x' \prec y'$), where one of the two inequalities is strict. We write $S \lhd S'$ for two sets of edges S and S' if $e \lhd e'$ for every $e \in S$ and $e' \in S'$.*

Note that two edges $e, e' \in B$ are comparable w.r.t. $\lhd$ if and only if e and e' are not in conflict.

Claim. The set $\mathcal{S}$ is totally ordered by $\lhd$. That is, for every two distinct connected components S and S' in $\mathcal{S}$ either $S \lhd S'$ or $S' \lhd S$.

We order elements of $\mathcal{S}$ (connected components of $\mathcal{C}[B]$) w.r.t. to the order $\lhd$: $S_1 \lhd \cdots \lhd S_r$. We note that sets $S_1, \ldots, S_r$ satisfy most properties we require in Theorem 4.1. Since sets $S_1, \ldots, S_r$ are ordered w.r.t. $\lhd$, endpoints of edges in sets S_i divide Q_1 and Q_2 into r consecutive segments. By Lemma 4.2, each

graph $\mathcal{C}[S \cup L_1 \cup L_2]$ is bipartite and therefore the graph $Q_1 \cup Q_2 \cup S \cup L_1 \cup L_2$ is planar. The only obstacle is that the number r of sets S_i can be arbitrarily large (it is not bounded by a function of g). To resolve this issue, we will join together some consecutive sets S_i and obtain the desired sets T_j. We do that using a simple greedy algorithm: We define numbers $t_0, t_1, t_2, \dots$ by induction. First, we let $t_0 = 0$. Let t_{q+1} be the largest t such that one of the following two conditions holds

1. the graph $\mathcal{C}[S_{t_q+1} \cup \cdots \cup S_t \cup L_1 \cup L_2]$ is bipartite;
2. all edges in $S_{t_q+1}, \dots, S_t$ are in conflict with some local edge $\hat{e} \in L_1 \cup L_2$.

We stop this procedure when we process all sets S_i. We obtain numbers $t_0, \dots, t_s$ (for some $s > 0$). Let $T_q = S_{t_{q-1}+1} \cup \cdots \cup S_{t_q}$ for every $q \in \{1, \dots, s\}$.

Since each set T_j is the union of consecutive sets S_i, we have that $T_1 \triangleleft T_2 \cdots \triangleleft \cdots \triangleleft T_s$. For every p consider all vertices in Q_1 incident to edges in T_p. Let Q_1^p be the segment between the leftmost and rightmost among such vertices; define Q_2^p similarly. Then $Q_1^1 \preceq Q_1^2 \preceq \cdots \preceq Q_1^s$, and $Q_2^1 \succeq Q_2^2 \succeq \cdots \succeq Q_1^s$. Note that Q_1^i and Q_1^{i+1} share at most one vertex and Q_1^i and Q_1^j are disjoint if $|i - j| > 1$.

Recall that we need to find a drawing of each graph $T_q \cup Q_1 \cup Q_2 \cup L_1 \cup L_2$ on a surface of genus $O(g^2)$ and prove that $s = O(g)$. Note that if $\mathcal{C}[S_{t_{q-1}+1} \cup \cdots \cup S_{t_q} \cup L_1 \cup L_2]$ is bipartite (the first stopping condition holds) then $T_q \cup Q_1 \cup Q_2 \cup L_1 \cup L_2$ is planar. So besides proving that $s = O(g)$, we only need to analyze the case when all edges in $S_{t_{q-1}+1}, \dots, S_{t_q}$ are in conflict with some local edge $\hat{e} \in L_1 \cup L_2$. We need the following definition that captures this case.

Definition 4.3 (Comb). *Let B_1 and B_2 be two bands. Let $B' \subset B_1 \cap B_2$. Suppose that all edges in B' are in conflict with a local edge $\hat{e} \in L_1$. Let φ_1 and φ_2 be canonical drawings of $H_1 = Q_1 \cup Q_2 \cup B' \cup L_1$ and $H_2 = Q_1 \cup Q_2 \cup B' \cup L_2$ respectively (as in Definition 3.2). Note that all edges in B' are drawn on one side of ℓ since all edges in B' are in conflict with $\hat{e}$; we assume w.l.o.g. that all edges in B' are drawn above ℓ.*

Then, we say that $((B_1, Q_1, \varphi_1), (B_2, Q_2, \varphi_2), B')$ is a comb *with* spine P*, or simply a comb, when P is clear from the context.*

Lemma 4.3. *For every set T_p, we have*

- *there is a canonical planar drawing φ of $Q_1 \cup Q_2 \cup L_1 \cup L_2 \cup T_p$, or*
- *there are drawings φ_1 and φ_2 such that $((B_1, Q_1, \varphi_1), (B_2, Q_2, \varphi_2), T_p)$ is a comb, or*
- *there are drawings φ_1 and φ_2 such that $((B_2, Q_2, \varphi_2), (B_1, Q_1, \varphi_1), T_p)$ is a comb.*

We present an algorithm that finds a drawing of a comb on a surface of genus $O(g^2)$ in the full version of this paper.

It remains to show $s = O(g)$. Roughly speaking, we observe that $T_i \cup T_{i+1} \cup Q_1 \cup Q_2 \cup L_1 \cup L_2$ is not planar because if it was planar our greedy algorithm would include T_{i+1} in T_i. Then we consider $s/2$ graphs $T_i \cup T_{i+1} \cup Q_1 \cup Q_2 \cup L_1 \cup L_2$ for $i \in \{1, 3, 5, \dots\}$. Each of them is not planar and there are $s/2$ of them. Note that

the union of k *disjoint* non-planar graphs has genus at least k. So we want to argue that $s/2 \le g$ as otherwise the union of graphs $T_i \cup T_{i+1} \cup Q_1 \cup Q_2 \cup L_1 \cup L_2$ would have genus at least $s/2 > g$, which would contradict the fact that the genus of G is g. However, this approach does not work as stated because graphs $T_i \cup T_{i+1} \cup Q_1 \cup Q_2 \cup L_1 \cup L_2$ share local edges. To overcome this obstacle, for each set T_i we define sets $\Lambda_1^i \subset L_1$ and $\Lambda_2^i \subset L_2$ such that $T_i \cup T_{i+1} \cup Q_1 \cup Q_2 \cup \Lambda_1^i \cup \Lambda_2^i$ is not planar and sets Λ_1^i and Λ_1^j do not intersect if $|i-j| > 2$. This resolves the problem in the argument we outlined above (we choose indices i with some sufficiently large constant gap). We present the formal proof in the full version of this paper. We obtain the following lemma, which concludes the proof of Theorem 4.2.

Lemma 4.4. *We have,* $s = O(\mathsf{genus}(G))$.

Proof of Theorem 4.2. We presented an algorithm that finds a drawing ψ_i of $Q_1 \cup Q_2 \cup T_i \cup L_1 \cup L_2$ on a surface of genus $O(g^2)$ for each i. Now to prove Theorem 4.2, we show how to combine all drawings ψ_i in one drawing ψ. We assume that the drawing of ℓ and $Q_1 \cup Q_2$ are the same in all drawings ψ_i (we can do that without loss of generality since all vertices in $Q_1 \cup Q_2$ are ordered w.r.t. $\prec$ in all drawings ψ_i).

First we take care of global edges in T_i. Consider the drawing ψ_i. Recall that if T_i is not a comb then the drawing ψ_i is planar. Otherwise, it is a drawing on a plane with attached handles. In the latter case, all handles are attached to the plane above or below either Q_1^i or Q_2^i. We make four puncture in the plane: one above Q_1^i (sufficiently far away from ℓ), one below Q_1^i, one above Q_2^i and one below Q_2^i. We attach a handle H^U between two punctures above ℓ, and another handle H^D between two punctures below ℓ. Now we redraw all global edges that go above ℓ on the handle H_U, and all edges that go below ℓ on the handle H_D. We cut the part of the plane that lies above and below Q_1^i and Q_2^i. We denote this part by $\mathcal{T}_i$. The boundary of $\mathcal{T}_i$ consists of 4 vertical lines that pass through the left end of Q_1^i, the right end of Q_1^i, the left end of Q_2^i, and the right end of Q_2^i. We combine these parts $\mathcal{T}_i$ together and get a surface $\mathcal{T}$ of genus at most $O(g^3)$; we do not identify boundaries of $\mathcal{T}_i$ except for points of ℓ that belong to boundaries of several sets $\mathcal{T}_i$ (at this point, surface might be disconnected). We also add line ℓ to $\mathcal{T}$. Now we partially define the drawing ψ on $\mathcal{T}$. We draw Q_1 and Q_2 in ψ in the same way as they are drawn in ψ_i. We draw each global edge $e \in T_i$ in the same way it is drawn in ψ_i (on $\mathcal{T}_i$). We perform this step for all i and obtain a drawing of all global edges.

Now we take care of local edges. Consider a local edge e whose drawing ψ_i partially lies in $\mathcal{T}_i$. We draw the segment of e that lies in $\mathcal{T}_i$ on $\mathcal{T}$ in the same way it is drawn on $\mathcal{T}_i$. It remains to draw missing segments of local edges and connect all segments together. We describe how we do that for edges in L_1; we process edges in L_2 in exactly the same way.

Consider two consecutive sets T_i and T_{i+1}. Let u be the the rightmost vertex of Q_1^i and v be the leftmost vertex of Q_{i+1}^i. Let e_i be a global edge in T_i incident on u, and e_{i+1} be a global edge in T_{i+1} incident on v, Let h_u be the line

perpendicular to u in $\mathcal{T}_i$ and h_v be the line perpendicular to v in $\mathcal{T}_{i+1}$. There are two possibilities. Either $u = v$ or $u \prec v$.

First, we consider the case $u = v$. Let A be the set of edges $e = (x, y)$ with $x \prec u$ and $u \prec y$. All edges in A are in conflict with both e and e'. Thus all edges in A are drawn on one side of ℓ in ψ_i and ψ_{i+1} (in particular, no two edges in A are in conflict). Consider all crossing points of edges in A and line h_u ordered descendingly by their distance from ℓ. Since the drawing of local edges in ψ_i is combinatorially planar, crossing points are ordered in the same way as corresponding edges ordered by $\prec$: if $e_1 \prec e_2$ then the crossing point of e_1 is further away from ℓ than the crossing point of e_2. Similarly, the crossing points of edges in A and line h_v are ordered in the same way as edges in A. Thus edges cross lines h_u and h_v in the same order. We attach a handle between $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ and then for every edge $e \in A$ draw a curve that connects the segment of e in $\mathcal{T}_i$ and the segment of e in $\mathcal{T}_{i+1}$.

Now consider the case when $u \prec v$. Let A be the set of edges $e = (x, y)$ with $x \prec u$ and $v \prec y$. We treat edges in A in exactly the same way as before; we attach one handle and connect segments of edges in A drawn on $\mathcal{T}_i$ and on $\mathcal{T}_{i+1}$. It remains to draw edges in the set $D = \{e = (x, y) : x \prec u \prec y \prec v$ or $u \prec x \prec v \prec y$ or $u \prec x \prec y \prec v\}$. Note that $A \prec D$ thus all crossing points of edges in D with h_u and h_v lie closer to ℓ then crossing points of A with h_u and h_v, respectively.

Denote the conflict graph $\mathcal{C}[D \cup \{e_i, e_{i+1}\}]$ by H.

Lemma 4.5. *The graph H is bipartite.*

Proof. Consider connected components of $\mathcal{C}[D]$. We show that there is at most one connected component C that is connected with both e_i and e_{i+1} in H. Assume to the contrary that there are two such connected components C_1 and C_2. Repeating the proof of Claim 4, we get that if two connected components C_1 and C_2 of $\mathcal{C}[D]$ are connected with e_i then either $C_1 \triangleleft C_2$ or $C_2 \triangleleft C_1$. Without loss of generality, $C_1 \triangleleft C_2$. Then for every edge $e = (x, y) \in C_1$ (with $x \prec y$) we have $x \preceq C_2$ and $C_2 \preceq y$. Thus $x \prec u$ and $v \prec y$, which contradicts to the fact that $e \notin A$.

Let C be a connected component of $\mathcal{C}[D]$. By Lemma 4.2, graphs $C \cup \{e_i\}$ and $C \cup \{e_{i+1}\}$ are bipartite. Since there is no edge between e_i and e_{i+1} in H, the graph $C \cup \{e_i\} \cup \{e_{i+1}\}$ is also bipartite. We color the graph H with two colors as follows. If there is a connected component of $\mathcal{C}[D]$ which is connected to both e_i and e_{i+1}, we first color it with 2 colors. Otherwise, we arbitrarily color nodes e_i and e_{i+1} of H. Every other connected component of $\mathcal{C}[D]$ is connected to at most one of nodes e_i and e_{i+1}. We color it with 2 colors so that its coloring agrees with the coloring of e_i and e_{i+1}. We obtain a valid 2-coloring of H.

Since H is bipartite there exist a canonical drawing γ of edges of D, in which all edges that are in conflict with e_i lie on one side of ℓ and all edges that are in conflict with e_{i+1} lie on one side of ℓ. We attach a slab $\mathcal{T}_{i,i+1}$ above and below the segment between u and v of Q_1 to $\mathcal{T}$. Then we draw segments of all edges in D on $\mathcal{T}_{i,i+1}$ in the same way they are drawn in γ. Now the leftmost vertex of

$\mathcal{T}_{i,i+1}$ is u and the rightmost vertex of $\mathcal{T}_{i,i+1}$ is v. So we can use the argument we used above to connect drawings of segments of $e \in D$ drawn on $\mathcal{T}_i$, $\mathcal{T}_{i,i+1}$, and $\mathcal{T}_{i+1}$.

5 Drawing a Hamiltonian Graph

In previous sections, we showed how to obtain $O(g^2)$ embeddings that are almost consistent. In the full version of this paper, we show how to resolve all remaining inconsistencies to obtain the final embedding.

Theorem 5.1 (Main result). *There exists a polynomial time algorithm which given a graph G of orientable genus g, and a Hamiltonian path in G, outputs a drawing of G into a surface of either orientable, or non-orientable genus $O(g^7)$.*

References

[CH11] Chimani, M., Hliněný, P.: A tighter insertion-based approximation of the crossing number. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 122–134. Springer, Heidelberg (2011)

[Chu11] Chuzhoy, J.: An algorithm for the graph crossing number problem. In: STOC, pp. 303–312 (2011)

[CKK97] Chen, J., Kanchi, S.P., Kanevsky, A.: A note on approximating graph genus. Inf. Process. Lett. 61(6), 317–322 (1997)

[CMS11] Chuzhoy, J., Makarychev, Y., Sidiropoulos, A.: On graph crossing number and edge planarization. In: SODA, pp. 1050–1069 (2011)

[CS] Chekuri, C., Sidiropoulos, A.: Approximating the genus of a graph. Manuscript

[FMR79] Filotti, I.S., Miller, G.L., Reif, J.H.: On determining the genus of a graph in $O(v^{O(g)})$ steps. In: STOC, pp. 27–37 (1979)

[GT01] Gross, J.L., Tucker, T.W.: Topological graph theory. Dover Publications (2001)

[Hat02] Hatcher, A.: Algebraic Topology. Cambridge University Press (2002)

[HT74] Hopcroft, J., Tarjan, R.: Efficient planarity testing. J. ACM 21(4), 549–568 (1974)

[KiMR08] Ken-ichi, Mohar, B., Reed, B.A.: A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In: FOCS, pp. 771–780 (2008)

[LR99] Leighton, F.T., Rao, S.: Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. J. ACM 46(6), 787–832 (1999)

[MC08] Chimani, M., Hliněný, P., Mutzel, P.: Approximating the crossing number of apex graphs. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 432–434. Springer, Heidelberg (2009)

[MM92] Malnic, A., Mohar, B.: Generating locally cyclic triangulations of surfaces. Journal of Combinatorial Theory, Series B 56(2), 147–164 (1992)

[Moh99] Mohar, B.: A linear time algorithm for embedding graphs in an arbitrary surface. SIAM J. Discrete Math. 12(1), 6–26 (1999)

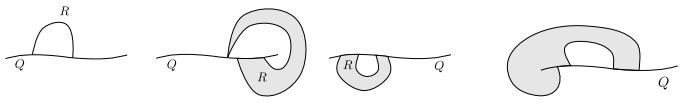
[Moh01] Mohar, B.: Face covers and the genus problem for apex graphs. Journal of Combinatorial Theory, Series B 82(1), 102–117 (2001)
[MT01] Mohar, B., Thomassen, C.: Graphs on surfaces. Johns Hopkins studies in the mathematical sciences. Johns Hopkins University Press (2001)
[PH10] Hlinený, P., Chimani, M.: Approximating the crossing number of graphs embeddable in any orientable surface. In: SODA, pp. 918–927 (2010)
[RS90] Robertson, N., Seymour, P.D.: Graph minors. VIII. A Kuratowski theorem for general surfaces. J. Comb. Theory, Ser. B 48(2), 255–288 (1990)
[Tho89] Thomassen, C.: The graph genus problem is np-complete. J. Algorithms 10(4), 568–576 (1989)
[Tho93] Thomassen, C.: Triangulating a surface with a prescribed graph. J. Comb. Theory, Ser. B 57(2), 196–206 (1993)
[Whi01] White, A.T.: Graphs of groups on surfaces: interactions and models. North-Holland mathematics studies. Elsevier (2001)

A Ribbons and Petals

Definition A.1 (Ribbon). *Let Q be a simple open curve in a surface $\mathcal{F}$. A set $A \subset \mathcal{F}$ is called a* Q-ribbon *if it satisfies one of the following conditions:*

(1) The set A is the image of a simple curve with endpoints $x, y \in Q$, and such that $A \cap Q = \{x, y\}$.
(2) Intuitively, A is a deformed triangle in $\mathcal{F}$ that intersects Q only on a vertex and its opposite edge. Formally, let T be 2-simplex, let a be a vertex of T, and let ℓ be the edge of T opposite to a. Then, there exists a continuous mapping $f : T \to \mathcal{F}$ such that f is a homeomorphism on $T \setminus \{a\}$, and on ℓ. Moreover, $f(a \cup \ell) \subseteq Q$, $f(T \setminus (a \cup \ell)) \cap Q = \emptyset$, and $f(T) = A$.
(3) Intuitively, A is a deformed rectangle in $\mathcal{F}$ that intersects Q only on two opposite edges. Formally, let $f : [0,1]^2 \to \mathcal{F}$ be a continuous mapping such that f is a homeomorphism on $(0,1)\times[0,1]$, on $\{0\}\times[0,1]$, and on $\{1\}\times[0,1]$. Moreover, $f((0,1)\times[0,1]) \cap Q = \emptyset$, $f(\{0,1\}\times[0,1]) \subseteq Q$, and $f([0,1]^2) = A$.

We say that the points $f((0,0))$, $f((0,1))$, $f((1,0))$, $f((1,1))$ are endpoints *of A. Figure 3(a) depicts examples of ribbons.*



(a) Examples of sets R that are Q-ribbons. (b) Not a Q-ribbon.

Fig. 3. Ribbons

Definition A.2 (Petal). *Let Q be a simple open curve in a surface $\mathcal{F}$. A set $X \subset \mathcal{F}$ is called a* Q-petal *if there exists a continuous mapping $f : [0,1]^2 \to \mathcal{F}$, so that f is a homeomorphism on $[0,1] \times (0,1]$, with $f((0,1] \times [0,1]) \cap Q = \emptyset$, $f(\{0\} \times [0,1]) \subseteq Q$, and $f([0,1]^2) = X$. We distinguish between the following three types of petals:*

(1) The Q-petal X is called single *if f is a homeomorphism on $[0,1]^2$.*
(2) The Q-petal X is called double *if it is not single, and if there exists $x \in [0,1]$, so that f is a homeomorphism on $\{0\} \times [0,x]$, and on $\{0\} \times [x,1]$.*
(3) The Q-petal X is called triple *if it is not single, and it is not double, and if there exist $x < y \in [0,1]$, so that f is a homeomorphism on $\{0\} \times [0,x]$, on $\{0\} \times [x,y]$, and on $\{0\} \times [y,1]$.*

We say that the points $f((0,0))$, and $f((0,1))$ are endpoints *of X. Figure 4 depicts examples of the three different types of petals.*

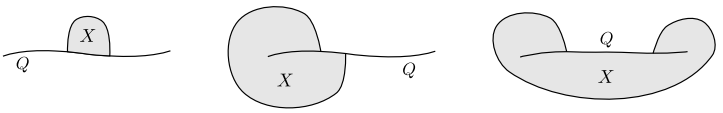


Fig. 4. From left to right: A single Q-petal, a double Q-petal, and a triple Q-petal X

Definition A.3 (Ribbon-petal covering). *Let G be a graph of genus g, and let P be a Hamiltonian path in G. Let φ be a drawing of G into a surface $\mathcal{S}$ of genus g. Let $\mathcal{X}$ be a collection of subsets of $\mathcal{S}$. Then, we say that $\mathcal{X}$ is a* ribbon-petal covering *for (G,P,φ), if the following conditions are satisfied:*

(i) Every $X \in \mathcal{X}$ is either a $\varphi(P)$-ribbon, or a $\varphi(P)$-petal.
(ii) For every $X, X' \in \mathcal{X}$, with $X \neq X'$, we have $X \cap X' \subseteq \varphi(P)$.
(iii) For every $e \in E(G) \setminus E(P)$, there exists $X \in \mathcal{X}$, such that $\varphi(e) \subseteq X$.

Lemma A.1. *Let G be a graph of genus g, and let P be a Hamiltonian path in G. Let φ be a drawing of G into a surface $\mathcal{S}$ of genus g. For every $e \in E(G) \setminus E(P)$, let P_e be the subpath of P between the endpoints of e, and define the cycle $C_e = P_e \cup \{e\}$. Let $E^* = \{e \in E(G) \setminus E(P) : \varphi(C_e)$ is non-contractible in $\mathcal{S}\}$. Let $X \subseteq E^*$, such that for every $e \neq e' \in X$, the cycles $\varphi(C_e)$ and $\varphi(C_{e'})$ are non-homotopic. Then, $|X| \leq 6g-3$.*

Proof. Let $\mathcal{S}'$ be the surface obtained from $\mathcal{S}$ by contracting $\varphi(P)$ into a single point. Let G' be the graph obtained from G by contracting P into a single vertex p. Note that G' contains a single vertex p, and multiple loops. Let φ' be the induced drawing of G' into G. Note that $\mathcal{S}'$ has genus g. For every $e \in X$, let C'_e be the loop in G' containing e and p. Moreover, for any $e \neq e' \in X$, the cycles $\varphi(C_e)$ and $\varphi(C_{e'})$ are homotopic in $\mathcal{S}$ if and only if $\varphi'(C'_e)$ and $\varphi'(C'_{e'})$ are homotopic in $\mathcal{S}'$. Therefore, by Lemma 2.1 we obtain that $|X| \leq 6g-3$.

Lemma A.2 (Existence of small ribbon-petal coverings). *Let G be a graph of genus g, and let P be a Hamiltonian path in G. Let φ be a drawing of G into a surface $\mathcal{S}$ of genus g. Then, there exists a ribbon-petal covering $\mathcal{X}$ for (G,P,φ), with $|\mathcal{X}| \leq 24g-12$.*

Proof. For every $e \in E(G) \setminus E(P)$, let P_e be the subpath of P between the endpoints of e, and define the cycle $C_e = P_e \cup \{e\}$. Let

$$E^* = \{e \in E(G) \setminus E(P) : \varphi(C_e) \text{ is non-contractible in } \mathcal{S}\}.$$

Consider the partition $E^* = Y_1 \cup \ldots \cup Y_k$, such that for every $i \in \{1, \ldots, k\}$, for every $e, e' \in Y_i$, the cycles $\varphi(C_e)$ and $\varphi(C_{e'})$ are homotopic, and for every $i \neq j \in \{1, \ldots, k\}$, for every $e \in Y_i$, $e' \in Y_j$, the cycles $\varphi(C_e)$ and $\varphi(C_{e'})$ are non-homotopic. By Lemma A.1, we have $k \leq 6g - 3$.

Let $i \in \{1, \ldots, k\}$. Since for all $e \in Y_i$, all the cycles $\varphi(C_e)$, $e \in Y_i$ are homotopic, it follows that there exists an ordering $e_1, \ldots, e_{k_i}$ of the edges in Y_i, and a continuous mapping $f_i : [0,1]^2 \to \mathcal{S}$, with $f_i([0,1] \times \{0,1\}) \subseteq \varphi(P)$, and moreover for every $e_j \in Y_i$, there exists $x_j \in [0,1]$, so that $f_i(x_j \times [0,1]) = \varphi(e_j)$. Since for every $i' \neq i \in \{1, \ldots, k\}$ and for every $e \in Y_i$, $e' \in Y_{i'}$, the cycles $\varphi(C_e)$ and $\varphi(C_{e'})$ are non-homotopic, it follows that we can pick the maps $f_1, \ldots, f_k$ so that for every $i \neq i' \in \{1, \ldots, k\}$ the sets $f_i([0,1]^2)$ and $f_{i'}([0,1]^2)$ have disjoint interiors. It therefore suffices to show how every set $A_i = f_i([0,1]^2)$ can be decomposed into at most three $\varphi(P)$-ribbons with disjoint interiors. To that end, we consider the following cases: Let a, b be the two endpoints of $\varphi(P)$.

(i) If $f_i([0,1] \times \{0\})$, and $f_i([0,1] \times \{1\})$ are both single points, then the set $f_i([0,1]^2)$ is clearly a $\varphi(P)$-ribbon.

(ii) If $f_i([0,1] \times \{0\})$ is a single point, and $f_i([0,1] \times \{1\})$ is not a single point, then we can pick f_i so that there exist at most two values $x < y \in [0,1]$ such that for every $z \in [0,1] \setminus \{x, y\}$, we have $f((z,1)) \notin \{a, b\}$. It follows that the sets $f_i([0,x] \times [0,1])$, $f_i([x,y] \times [0,1])$, $f_i([y,1] \times [0,1])$ are the desired $\varphi(P)$-ribbons.

(iii) If $f_i([0,1] \times \{0\})$ is not a single point, and $f_i([0,1] \times \{1\})$ is a single point, we can decompose $f_i([0,1]^2)$ into at most three $\varphi(P)$-ribbons as in the previous case.

(iv) If $f_i([0,1] \times \{0\})$ is not a single point, and $f_i([0,1] \times \{1\})$ is not a single point, then we can pick f_i so that there exist at most two values $x < y \in [0,1]$ such that for every $z \in [0,1] \setminus \{x, y\}$, we have $f((z,1)) \notin \{a, b\}$, and $f((z,0)) \notin \{a, b\}$. It follows that the sets $f_i([0,x] \times [0,1])$, $f_i([x,y] \times [0,1])$, $f_i([y,1] \times [0,1])$ are the desired $\varphi(P)$-ribbons.

We perform the above decomposition to each set $f_i([0,1]^2)$, $i \in \{1, \ldots, k\}$.

For every $e \in E^*$, the cycle $\varphi(C_e)$ is contractible in $\mathcal{S}$. Therefore, it bounds a disk $D_e \subset \mathcal{S}$, and this disk is a $\varphi(P)$-petal. Let $e, e' \in E^*$. Since $\varphi(e) \cap \varphi(e') \subset \varphi(P)$, we have that either $D_e \subset D_{e'}$, or $D_{e'} \subset D_e$, or $D_e \cap D_{e'} \subset \varphi(P)$. It follows that we can cover all the disks $\{D_e\}_{e \in E(G) \setminus E^*}$ by using at most one $\varphi(P)$-petal between every two consecutive ribbons on every side of $\varphi(P)$, and possibly one more $\varphi(P)$-petal for every one of the two endpoints of $\varphi(P)$. Since every ribbon intersects $\varphi(P)$ in at most two segments, in total we obtain a collection of at most k petals satisfying the assertion.

A Local Computation Approximation Scheme to Maximum Matching

Yishay Mansour* and Shai Vardi**

School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
{mansour,shaivar1}@post.tau.ac.il

Abstract. We present a polylogarithmic local computation matching algorithm which guarantees a $(1-\epsilon)$-approximation to the maximum matching in graphs of bounded degree.

Keywords: Local Computation Algortithms, Sublinear Algorithms, Approximation Algorithms, Maximum Matching.

1 Introduction

Finding *matchings* - sets of vertex disjoint edges in a graph - has been an important topic of research for computer scientists for over 50 years. Of particular importance is finding *maximum* matchings - matchings of maximal cardinality. Algorithms that find a maximum matching have many applications in computer science; in fact, their usefulness extends far beyond the boundaries of computer science - to disciplines such as economics, biology and chemistry.

The first works on matching were based on unweighted bipartite graphs (representing problems such as matching men and women). Hall's marriage theorem [6] gives a necessary and sufficient condition for the existence of a perfect matching[1]. The efficient algorithms for the weighted bipartite matching problem date back to the Hungarian method [12,18]. In this work we focus on maximum matchings in general unweighed graphs. Berge [3] proved that a matching is a maximum matching if and only if the graph has no augmenting paths with respect to the matching. Edmonds used augmenting paths to find a maximum matching in his seminal work [5], in which he showed that a maximum matching can be found in polynomial time. Much work on matching been done since (e.g., [7,9,16,17]). Our work uses ideas from Hopcroft and Karp's algorithm for finding maximal matching in bipartite graphs [9], which runs in time $O(n^{2.5})$.

Local computation algorithms (LCAs) [20] consider the scenario in which we must respond to queries (regarding a feasible solution) quickly and efficiently, yet we never need the entire solution at once. The replies to the queries need to be *consistent*; namely, the responses to all possibly queries combine to a single feasible solution. For example,

* Supported in part by the Google Inter-university center for Electronic Markets and Auctions, by a grant from the Israel Science Foundation, by a grant from United States-Israel Binational Science Foundation (BSF), and the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11).

** Supported in part by the Google Europe Fellowship in Game Theory.

[1] A perfect matching includes all the nodes of a bipartite graph.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 260–273, 2013.

an LCA for matching in a graph G, receives an *edge-query* for an edge $e \in G$ and replies "yes" if and only if e is part of the matching. The replies to all the possible edge queries define a matching in the graph.

In this work we present a *local computation approximation scheme* to maximum matching. Specifically, we present an LCA such that for any $\epsilon > 0$, the edge-query replies comprise a matching that is a $(1-\epsilon)$-approximation to the maximum matching. Our LCA requires $O(\log^3 n)$ space, and with probability at least $1 - 1/n^2$, for any edge-query, it runs in time $O(\log^4 n)$. To the best of our knowledge, this is the first local computation approximation algorithm for a problem which provably does not have an LCA.

Related Work. In the distributed setting, Itai and Israeli [10] showed a randomized algorithm which computes a maximal matching (which is a $1/2$-approximation to the maximum matching) and runs in $O(\log n)$ time with high probability. This result has been improved several times since (e.g., [4,8]); of particular relevance is the approximation scheme of Lotker et al. [13], which, for every $\epsilon > 0$, computes a $(1-\epsilon)$-approximation to the maximum matching in $O(\log n)$ time. Kuhn et al., [11] proved that any distributed algorithm, randomized or deterministic, requires (in expectation) $\Omega(\sqrt{\log n / \log\log n})$ time to compute a $\Theta(1)$-approximation to the maximum matching, even if the message size is unbounded.

Rubinfeld et al., [20] showed how to transform distributed algorithms to LCAs, and gave LCAs for several problems, including maximal independent set and hypergraph 2-coloring. Unfortunately, their method bounds the running time of the transformed algorithm exponentially in the running time of the distributed algorithm. Therefore, distributed algorithms for approximate maximum matching cannot be (trivially) transformed to LCAs using their technique.

Query trees model the dependency of queries on the replies to other queries, and were introduced in the local setting by Nguyen and Onak [19]. If a random permutation of the vertices is generated, and a sequential algorithm is simulated on this order, the reply to a query on vertex v depends only on the replies to queries on the neighbors of v which come before it in the permutation. Alon et al., [2] showed that if the running time of an algorithm is $O(f(n))$, where f is polylogarithmic in n, a $1/n^2$ - almost $f(n)$-independent ordering on the vertices can be generated in time $O(f(n)\log^2 n)$, thus guaranteeing the polylogarithmic space bound of any such algorithm. Mansour et al., [14] showed that the size of the query tree can be bounded, with high probability, by $O(\log n)$, for graphs of bounded degree. They also showed that it is possible to transform many on-line algorithms to LCAs. One of their examples is an LCA for maximal matching, which immediately gives a $1/2$-approximation to the maximum matching. In a recent work, [15], LCAs were presented for mechanism design problems. One of their impossibility results shows that any LCA for maximum matching requires $\Omega(n)$ time.

2 Notation and Preliminaries

2.1 Graph Theory

For an undirected graph $G = (V, E)$, a *matching* is a subset of edges $M \subseteq E$ such that no two edges $e_1, e_2 \in M$ share a vertex. We denote by M^* a matching of maximum

cardinality. An *augmenting path* with respect to a matching M is a simple path whose endpoints are *free* (i.e., not part of any edge in the matching M), and whose edges alternate between $E \setminus M$ and M. A set of augmenting paths P is *independent* if no two paths $p_1, p_2 \in P$ share a vertex.

For sets A and B, we denote $A \oplus B \stackrel{def}{=} (A \cup B) \setminus (A \cap B)$. An important observation regarding augmenting paths and matchings is the following.

Observation 1. *If M is a matching and P is an independent set of augmenting paths, then $M \oplus P$ is a matching of size $|M| + |P|$.*

A vertex $u \in V$ is a *neighbor* of vertex $v \in V$ if $(u, v) \in E$. Let $N(v)$ denote the set of neighbors of v, i.e., $N(v) = \{u : (v, u) \in E\}$. We assume that we have direct access both to $N(v)$ and to individual edges.

An independent set (IS) is a subset of vertices $W \subseteq V$ with the property that for any $u, v \in W$ we have $(u, v) \notin E$, namely, no two vertices $u, v \in W$ are neighbors in G. The IS is *maximal* (denoted by MIS) if no other vertices can be added to it without violating the independence property.

2.2 Local Computation Algorithms

We use the following model of local computation algorithms (LCAs)[20].[2] A $(t(n), s(n), \delta(n))$ - *local computation algorithm* $\mathcal{LA}$ for a computational problem is a (randomized) algorithm which receives an input of size n, and a query x. Algorithm $\mathcal{LA}$ uses at most $s(n)$ memory, and with probability at least $1 - \delta(n)$, it replies to any query x in time $t(n)$. The algorithm must be *consistent*, that is, the replies to all of the possible queries combine to a single feasible solution to the problem.

2.3 Query Trees

Let $G = (V, E)$ be a graph of bounded degree d. A real number $r(v) \in [0, 1]$ is assigned independently and uniformly at random to every vertex v in the graph. We refer to this random number as the *rank* of v. Each vertex in the graph G holds an input $x(v) \in R$, where the range R is some finite set. A randomized Boolean function F is defined inductively on the vertices in the graph such that $F(v)$ is a function of the input $x(v)$ at v as well as the values of F at the neighbors w of v for which $r(w) < r(v)$.

We would like to upper bound the number of queries that are needed to be made vertices in the graph in order to compute $F(v_0)$ for any vertex $v_0 \in G$. We turn to the simpler task of bounding the size of a certain d-regular tree, which is an upper bound on the number of queries. Consider an infinite d-regular tree $\mathcal{T}$ rooted at v_0. Each node w in $\mathcal{T}$ is assigned independently and uniformly at random a distinct real number $r(w) \in [0, 1]$. For every node $w \in \mathcal{T}$ other than v_0, let $\texttt{parent}(w)$ denote the parent

[2] Our model differs slightly from the model of [20] in that their model requires that the LCA *always* obeys the time and space bounds, and returns an error with some probability. It is easy to see that any algorithm which conforms to our model can be modified to conform to the model of [20] by forcing it to return an error if the time or space bound is violated.

node of w. We grow a (possibly infinite) subtree T of $\mathcal{T}$ rooted at v as follows: a node w is in the subtree T if and only if $\texttt{parent}(w)$ is in T and $r(w) < r(\texttt{parent}(w))$. We keep growing T in this manner such that a node $w' \in T$ is a leaf node in T if the ranks of its d children are all larger than $r(w')$. We call the random tree T constructed in this way a *query tree* and we denote by $|T|$ the random variable that corresponds to the size of T. Note that $|T|$ is an upper bound on the number of queries.

If the reply to a query q depends (only) on the replies to a set of queries, Q, we call Q the set of *relevant* queries with respect to q.

2.4 Random Orders

Let $[n]$ denote the set $\{1, \ldots, n\}$.

A distribution $D : \{0,1\}^n \to \mathbb{R}^{\geq 0}$ is *k-wise independent* if, when D is restricted to any index subset $S \subset [n]$ of size at most k, the induced distribution over S is the uniform distribution.

A random ordering $D_{\mathbf{r}}$ induces a probability distribution over permutations of $[n]$. It is said to *ϵ-almost k-wise independent* if for any subset $S \subset [n]$ of size at most k, the variation distance between the distribution induced by $D_{\mathbf{r}}$ on S and a uniform permutation over S is at most ϵ. We use the following Theorem from [2].

Theorem 2 ([2]). *Let $n \geq 2$ be an integer and let $2 \leq k \leq n$. Then there is a construction of $\frac{1}{n^2}$-almost k-wise independent random ordering over $[n]$ whose seed length is $O(k \log^2 n)$.*

We provide a short, intuitive explanation of the construction. We can construct n k-wise independent random variables $Z = (z_1, \ldots, z_n)$, using a seed of length $k \log n$ (see [1]). We generate $4 \log n$ independent copies of k-wise independent random variables, $Z_1, \ldots Z_{4 \log n}$. For $i \in [n]$, taking the i-th bit of each $Z_j, 1 \leq j \leq 4 \log n$ makes for a random variable $r(i) \in \{0,1\}^{4 \log n}$, which can be expressed as an integer in $\{0, 1, \ldots, n^4 - 1\}$. The order is induced by r (u comes before v in the order if $r(u) < r(v)$). The probability that there exists $u, v \in [n]$ such that $r(u) = r(v)$ is at most $1/n^2$, hence the ordering is $1/n^2$-almost k-wise independent.

3 Approximate Maximum Matching

We present a local computation approximation scheme for maximum matching: We show an LCA that, for any $\epsilon > 0$, computes a maximal matching which is a $(1 - \epsilon)$-approximation to the maximum matching.

Our main result is the following theorem:

Theorem 3. *Let $G = (V, E)$ be a graph of bounded degree d. Then there exists an $(O(\log^4 n), O(\log^3 n), 1/n)$ - LCA that, for every $\epsilon > 0$, computes a maximal matching which is a $(1 - \epsilon)$-approximation to the maximum matching.*

Our algorithm is, in essence, an implementation of the abstract algorithm of Lotker et al., [13]. Their algorithm, relies on several interesting results due to Hopcroft and Karp [9]. First, we briefly recount some of these results, as they are essential for the understanding of our algorithm.

3.1 Distributed Maximal Matching

While the main result of Hopcroft and Karp [9] is an improved matching algorithm for bipartite graphs, they show the following useful lemmas for general graphs. The first lemma shows that if the current matching has augmenting paths of length at least ℓ, then using a maximal set of augmenting paths of length ℓ will result in a matching for which the shortest augmenting path is strictly longer than ℓ. This gives a natural progression for the algorithm.

Lemma 4. *[9] Let $G = (V, E)$ be an undirected graph, and let M be some matching in G. If the shortest augmenting path with respect to M has length ℓ and Φ is a maximal set of independent augmenting paths of length ℓ, the shortest augmenting path with respect to $M \oplus \Phi$ has length strictly greater than ℓ.*

The second lemma shows that if there are no short augmenting paths then the current matching is approximately optimal.

Lemma 5. *[9] Let $G = (V, E)$ be an undirected graph. Let M be some matching in G, and let M^* be a maximum matching in G. If the shortest augmenting path with respect to M has length $2k - 1 > 1$ then $|M| \geq (1 - 1/k)|M^*|$.*

Lotker et al., [13] gave the following abstract approximation scheme for maximal matching in the distributed setting.[3] Start with an empty matching. In stage $\ell = 1, 3, \ldots, 2k-1$, add a maximal independent collection of augmenting paths of length ℓ. For $k = \lceil 1/\epsilon \rceil$, by Lemma 5, we have that the matching M_ℓ is a $(1 - \epsilon)$-approximation to the maximum matching.

In order to find such a collection of augmenting paths of length ℓ, we need to define a conflict graph:

Definition 6. *[13] Let $G = (V, E)$ be an undirected graph, let $M \subseteq E$ be a matching, and let $\ell > 0$ be an integer. The ℓ-conflict graph with respect to M in G, denoted $C_M(\ell)$, is defined as follows. The nodes of $C_M(\ell)$ are all augmenting paths of length ℓ, with respect to M, and two nodes in $C_M(\ell)$ are connected by an edge if and only if their corresponding augmenting paths intersect at a vertex of G.*[4]

We present the abstract distributed algorithm of [13], **AbstractDistributedMM**.

Note that for M_ℓ, the minimal augmenting path is of length at least $\ell+2$. This follows since $\Phi(M_{\ell-2})$ is a maximal independent set of augmenting paths of length ℓ. When we add $\Phi(M_{\ell-2})$ to $M_{\ell-2}$, to get M_ℓ, by Lemma 4 all the remaining augmenting paths are of length at least $\ell + 2$ (recall that augmenting paths have odd lengths).

Lines 4 - 7 do the task of computing M_ℓ as follows: the conflict graph $C_{M_{\ell-2}}(\ell)$ is constructed and an MIS, $\Phi(M_{\ell-2})$, is found in it. $\Phi(M_{\ell-2})$ is then used to augment $M_{\ell-2}$, to give M_ℓ.

[3] This approach was first used by Hopcroft and Karp in [9]; however, they only applied it efficiently in the bipartite setting.

[4] Notice that the nodes of the conflict graph represent *paths* in G. Although it should be clear from the context, in order to minimize confusion, we refer to a vertex in G by *vertex*, and to a vertex in the conflict graph by *node*.

Algorithm 1 - **AbstractDistributedMM** - Abstract distributed algorithm with input $G = (V, E)$ and $\epsilon > 0$

1: $M_{-1} \leftarrow \emptyset$ ▷ M_{-1} is the empty matching
2: $k \leftarrow \lceil 1/\epsilon \rceil$
3: **for** $\ell \leftarrow 1, 3, \ldots, 2k-1$, **do**
4: Construct the conflict graph $C_{M_{\ell-2}}(\ell)$
5: Let $\mathcal{I}$ be an MIS of $C_{M_{\ell-2}}(\ell)$
6: Let $\Phi(M_{\ell-2})$ be the union of augmenting paths corresponding to $\mathcal{I}$
7: $M_\ell \leftarrow M_{\ell-2} \oplus \Phi(M_{\ell-2})$ ▷ M_ℓ is matching at the end of phase ℓ
8: **end for**
9: Output M_ℓ ▷ M_ℓ is a $(1 - \frac{1}{k+1})$-approximate maximum matching

We would like to simulate this algorithm locally. Our main challenge is to simulate Lines 4 - 7 without explicitly constructing the entire conflict graph $C_{M_{\ell-2}}(\ell)$. To do this, we will simulate an on-line MIS algorithm.

3.2 Local Simulation of the On-Line Greedy MIS Algorithm

In the on-line setting, the vertices arrive in some unknown order, and **GreedyMIS** operates as follows: Initialize the set $I = \emptyset$. When a vertex v arrives, **GreedyMIS** checks whether any of v's neighbors, $N(v)$, is in I. If none of them are, v is added to I. Otherwise, v is not in I. (The pseudocode for **GreedyMIS** can be found in the full version of the paper.)

In order to simulate **GreedyMIS** locally, we first need to fix the order (of arrival) of the vertices, π. If we know that each query depends on at most k previous queries, we do not need to explicitly generate the order π on all the vertices (as this would take at least linear time). By Theorem 2, we can produce a $\frac{1}{n^2}$-almost-k-wise independent random ordering on the edges, using a seed, s, of length $O(k \log^2 n)$.

Technically, this is done as follows. Let r be a function $r : (v, s) \to [cn^4]$, for some constant c.[5] The vertex order π is determined as follows: vertex v appears before vertex u in the order π if $r(v, s) < r(u, s)$. Let $G' = (V', E')$ be the subgraph of G induced by the vertices $V' \subseteq V$; we denote by $\pi(G', s)$ the partial order of π on V'. Note that we only need to store s in the memory: we can then compute, for any subset V', the induced order of their arrival.

When simulating **GreedyMIS** on the conflict graph $C_M(\ell) = (V_{C_M}, E_{C_M})$, we only need a subset of the nodes, $V' \subseteq V_{C_M}$. Therefore, there is no need to construct $C_M(\ell)$ entirely; only the relevant subgraph need be constructed. This is the main observation which allows us to bound the space and time required by our algorithm.

[5] Alternately, we sometimes view r as a function $r : (v, s) \to [0, 1]$: Let r' be a function $r' : (v, s) \to [cn^4]$, and let $f : [cn^4] \to [0, 1]$ be a function that maps each $x \in [cn^4] - \{1\}$ uniformly at random to the interval $((x-1)/cn^4, x/cn^4]$, and f maps 1 uniformly at random to the interval $[0, 1/cn^4]$. Then set $r(v, s) = f(r'(v, s))$.

3.3 LCA for Maximal Matching

We present our algorithm for maximal matching - **LocalMM**, and analyze it. (The pseudocode for **LocalMM** can be found in the full version of the paper.) In contrast to the distributed algorithm, which runs iteratively, **LocalMM** is recursive in nature. In each iteration of **AbstractDistributedMM**, a maximal matching M_ℓ, is computed, where M_ℓ has no augmenting path of length less than ℓ. We call each such iteration a *phase*, and there are a total of k phases: $1, 3, \ldots 2k-1$. To find out whether an edge $e \in E$ is in M_ℓ, we recursively compute whether it is in $M_{\ell-2}$ and whether it is in $\Phi(M_{\ell-2})$, a maximal set of augmenting paths of length ℓ. We use the following simple observation to determine whether $e \in M_\ell$. The observation follows since $M_\ell \leftarrow M_{\ell-2} \oplus \Phi(M_{\ell-2})$.

Observation 7. *$e \in M_\ell$ if and only if it is in either in $M_{\ell-2}$ or in $\Phi(M_{\ell-2})$, but not in both.*

Recall that **LocalMM** receives an edge $e \in E$ as a query, and outputs "yes/no". To determine whether $e \in M_{2k-1}$, it therefore suffices to determine, for $\ell = 1, 3, \ldots, 2k-3$, whether $e \in M_\ell$ and whether $e \in \Phi(M_\ell)$.

We will outline our algorithm by tracking a single query. (The initialization parameters will be explained at the end.) When queried on an edge e, **LocalMM** calls the procedure IsInMatching with e and the number of phases k. For clarity, we sometimes omit some of the parameters from the descriptions of the procedures.

Procedure IsInMatching determines whether an edge e in in the matching M_ℓ. To determine whether $e \in M_\ell$, IsInMatching recursively checks whether $e \in M_{\ell-2}$, by calling IsInMatching$(\ell-2)$, and whether e is in some path in the MIS $\Phi(M_{\ell-2})$ of $C_{M_{\ell-2}}(\ell)$. This is done by generating all paths p of length ℓ that include e, and calling IsPathInMIS(p) on each. IsPathInMIS(p) checks whether p is an augmenting path, and if so, whether it in the independent set of augmenting paths. By Observation 7, we can compute whether e is in M_ℓ given the output of the calls.

Procedure IsPathInMIS receives a path p and returns whether the path is in the MIS of augmenting paths of length ℓ. The procedure first computes all the relevant augmenting paths (relative to p) using RelevantPaths. Given the set of relevant paths (represented by nodes) and the intersection between them (represented by edges) we simulate **GreedyMIS** on this subgraph. The resulting independent set is a set of independent augmenting paths. We then just need to check if the path p is in that set.

Procedure RelevantPaths receives a path p and returns all the relevant augmenting paths relative to p. The procedure returns the subgraph of $C_{M_{\ell-2}}(\ell), C = (V_C, E_C)$, which includes p and all the relevant nodes. These are exactly the nodes needed for the simulation of **GreedyMIS**, given the order induced by seed s_ℓ. The set of augmenting paths V_C is constructed iteratively, by adding an augmenting path q if it intersects some path $q' \in V_C$ and arrives before it (i.e., $r(q, s_\ell) < r(q', s_\ell)$). In order to determine whether to add path q to V_C, we need first to test if q is indeed a valid augmenting path, which is done using IsAnAugmentingPath.

Procedure IsAnAugmentingPath tests if a given path p is an augmenting path. It is based on the following observation.

Observation 8. *For any graph $G = (V, E)$, let M be a matching in G, and let $p = e_1, e_2, \ldots, e_\ell$ be a path in G. Path p is an augmenting path with respect to M if and only if all odd numbered edges are not in M, all even numbered edges are in M, and both the vertices at the ends of p are free.*

Given a path p of length ℓ, to determine whether $p \in C_{M_{\ell-2}}(\ell)$, IsAnAugmentingPath($\ell$) determines, for each edge in the path, whether it is in $M_{\ell-2}$, by calling IsInMatching($\ell - 2$). It also checks whether the end vertices are free, by calling Procedure IsFree(ℓ), which checks, for each vertex, if any of its adjacent edges are in $M_{\ell-2}$. From Observation 8, IsAnAugmentingPath(ℓ) correctly determines whether p is an augmenting with respect to $M_{\ell-2}$.

We end by describing the initialization procedure Initialize, which is run only once, during the first query. The procedure sets the number of phases to $\lceil 1/\epsilon \rceil$. It is important to set a different seed s_ℓ for each phase ℓ, since the conflict graphs are unrelated (and even the size of the description of each node, a path of length ℓ, is different). The lengths of the k seeds, $s_1, s_3, \ldots, s_{2k-1}$, determine our memory requirement.

3.4 Bounding the Complexity

In this section we prove Theorem 3. We start with the following observation:

Observation 9. *In any graph $G = (V, E)$ with bounded degree d, each edge $e \in E$ can be part of at most $\ell(d-1)^{\ell-1}$ paths of length ℓ. Furthermore, given e, it takes at most $O(\ell(d-1)^{\ell-1})$ time to find all such paths.*

Proof. Consider a path $p = (e_1, e_2, \ldots, e_\ell)$ of length ℓ. If p includes the edge e, then e can be in one of the ℓ positions. Given that $e_i = e$, there are at most $d - 1$ possibilities for e_{i+1} and for e_{i-1}, which implies at most $(d-1)^{\ell-1}$ possibilities to complete the path to be of length ℓ. □

Observation 9 yields the following corollary.

Corollary 10. *The ℓ-conflict graph with respect to any matching M in $G = (V, E)$, $C_M(\ell)$, consists of at most $\ell(d-1)^{\ell-1}|E| = O(|V|)$ nodes, and has maximal degree at most $d(\ell+1)\ell(d-1)^{\ell-1}$.*

Proof. (For the degree bound.) Each path has length ℓ, and therefore has $\ell + 1$ vertices. Each vertex has degree at most d, which implies $d(\ell+1)$ edges. Each edge is in at most $\ell(d-1)^{\ell-1}$ paths. □

Our main task will be to compute a bound on the number of recursive calls. First, let us summarize a recursive call. The only procedure whose runtime depends on the order induced by s_ℓ is RelevantPaths, which depends on the number of vertices V_C (which is a random variable depending of the seed s_ℓ). To simplify the notation we define the random variable $X_\ell = d(\ell+1)\ell(d-1)^{\ell-1}|V_C|$. Technically, **GreedyMIS** also depends on V_C, but its running time is dominated by the running time of RelevantPaths.

Calling procedure	Called Procedures
IsInMatching(ℓ)	$1\times$ IsInMatching($\ell-2$) and $\ell(d-1)^{\ell-1}\times$ IsPathInMIS(ℓ)
IsPathInMIS(ℓ)	$1\times$ RelevantPaths(ℓ) and $1\times$ **GreedyMIS**
RelevantPaths(ℓ)	$X_\ell\times$ IsAnAugmentingPath(ℓ)
IsAnAugmentingPath(ℓ)	$\ell\times$ IsInMatching($\ell-2$) and $2\times$ IsFree(ℓ)
IsFree(ℓ)	$(d-1)\times$ IsInMatching($\ell-2$)

From the table, it is easy to deduce the following proposition.

Proposition 11. IsAnAugmentingPath(ℓ) *generates at most* $\ell+2(d-1)$ *calls to* IsInMatching($\ell-2$)*, and therefore at most* $(\ell+2d-2)\cdot\ell(d-1)^{\ell-1}$ *calls to* IsPathInMIS($\ell-2$)*.*

We would like to bound X_ℓ, the number of calls to IsAnAugmentingPath(ℓ) during a single execution of IsPathInMIS(G,p,ℓ,S). We require the following theorem, the proof of which appears in Section 4.

Theorem 12. *For any infinite query tree* T *with bounded degree* d*, there exists a constant* c*, which depends only on* d*, such that for any large enough* $N>0$*,*

$$Pr[|T|>N]\leq e^{-cN}.$$

As a query tree T of bounded degree $D=d(\ell+1)\ell(d-1)^{\ell-1}$ is an upper bound to X_ℓ (by Corollary 10, D is an upper bound on the degree of $C_{M_{\ell-2}}(\ell)$), we have the following corollary to Theorem 12.

Corollary 13. *There exists an absolute constant* c*, which depends only on* d*, such that for any large enough* $N>0$*,*

$$Pr[X_\ell>N]\leq e^{-cN}.$$

Denote by f_ℓ the number of calls to IsAnAugmentingPath(ℓ) during one execution of **LocalMM**. Let $f=\sum_{\ell=1}^{2k-1}f_\ell$.[6] The base cases of the recursive calls **LocalMM** makes are IsAnAugmentingPath(1) (which always returns TRUE). As the execution of each procedure of **LocalMM** results in at least one call to IsAnAugmentingPath, f (multiplied by some small constant) is an upper bound to the total number of computations made by **LocalMM**.

We state the following proposition, the proof of which appears in Section 4.

Proposition 14 *Let* W_i *be a random variable. Let* $z_1,z_2,\ldots z_{W_i}$ *be random variables, (some possibly equal to* 0 *with probability* 1*). Assume that there exist constants* c *and* μ *such that for all* $1\leq j\leq W_i$*,* $Pr[z_j\geq\mu N]\leq e^{-cN}$*, for all* $N>0$*. Then there exist constants* μ_i *and* c_i'*, which depend only on* d*, such that for any* $q_i>0$*,*

$$Pr[\sum_{j=1}^{W_i}z_j\geq\mu_i q_i|W_i\leq q_i]\leq e^{-c_i'q_i}.$$

Using Proposition 14, we prove the following:

[6] For all even ℓ, let $f_\ell=0$.

Proposition 15. *For every $1 \leq \ell \leq 2k-1$, there exist constants μ_ℓ and c_ℓ, which depend only on d and ϵ, such that for any large enough $N > 0$*

$$Pr[f_\ell > \mu_\ell N] \leq e^{-c_\ell N}.$$

Proof. The proof is by induction. For the base of the induction, we have, from Corollary 13, that there exists an absolute constant c_{2k-1}, which depends only on d, such that for any large enough $N > 0$, $Pr[X_{2k-1} > N] \leq e^{-c_{2k-1}N}$. Assume that the proposition holds for $\ell = 2k-1, 2k-3, \ldots \ell$, and we show that it holds for $\ell - 2$.

Let $b_\ell = (\ell + 2d - 2) \cdot \ell(d-1)^{\ell-1}$. From Proposition 11, we have that each call to IsAnAugmentingPath(ℓ) generates at most b_ℓ calls to IsPathInMIS($\ell - 2$), and hence $b_\ell \cdot X_{\ell-2}$ calls to IsAnAugmentingPath($\ell - 2$). From Corollary 13, we have that there exists an absolute constant c, which depends only on d, such that for any large enough $N > 0$,

$$Pr[X_{\ell-2} > N] \leq e^{-cN}.$$

Setting $W_\ell = b_\ell f_\ell$, $f_{\ell-2} = \sum_{j=1}^{W_i} z_j$, $q_i = b_\ell \mu_\ell y_\ell$, and $\mu_i = \mu_{\ell-2}/b_\ell\mu_\ell$, and letting $c'_i = c'_\ell / b_\ell \mu_\ell$ in Proposition 14 implies the following:

$$Pr[f_{\ell-2} > \mu_{\ell-2} y_\ell | f_\ell \leq \mu_\ell y_\ell] \leq e^{-c'_\ell y_\ell}. \tag{1}$$

We have

$$\begin{aligned} Pr[f_{\ell-2} > \mu_{\ell-2}N] =& Pr[f_{\ell-2} > \mu_{\ell-2}N | f_\ell \leq \mu_\ell N] \cdot Pr[f_\ell \leq \mu_\ell N] \\ &+ Pr[f_{\ell-2} > \mu_{\ell-2}N | f_\ell > \mu_\ell N] \cdot Pr[f_\ell > \mu_\ell N] \\ \leq& Pr[f_{\ell-2} > \mu_{\ell-2}N | f_\ell \leq \mu_\ell N] + Pr[f_\ell > \mu_\ell N] \\ \leq& e^{-c'_\ell N} + e^{-c_\ell N} \\ =& e^{-c_{\ell-2}N}, \end{aligned} \tag{2}$$

where Inequality 2 stems from Inequality 1 and the induction hypothesis. □

Taking a union bound over all k levels immediately gives

Lemma 16. *There exists a constant c, which depends only on d and ϵ, such that*

$$Pr[f > c \log n] \leq 1/n^2.$$

Proof (Proof of Theorem 3). Using Lemma 16, and taking a union bound over all possible queried edges gives us that with probability at least $1 - 1/n$, **LocalMM** will require at most $O(\log n)$ queries. Therefore, for each execution of **LocalMM**, we require at most $O(\log n)$-independence for each conflict graph, and therefore, from Theorem 2, we require $\lceil 1/\epsilon \rceil$ seeds of length $O(log^3 n)$, which upper bounds the space required by the algorithm. The time required is upper bound by the time required to compute $r(p)$ for all the required nodes in the conflict graphs, which is $O(\log^4 n)$. □

4 Combinatorial Proofs

We want to bound the total number of queries required by Algorithm **LocalMM**.

Let T be a d-regular query tree. As in [2,14], we partition the interval [0,1] into $L \geq d+1$ sub-intervals: $I_i = (1 - \frac{i}{L+1}, 1 - \frac{i-1}{L+1}]$, for $i = 1, 2, \cdots, L$ and $I_{L+1} = [0, \frac{1}{L+1}]$. We refer to interval I_i as *level* i. A vertex $v \in T$ is said to be on level i if $r(v) \in I_i$. Assume the worst case, that for the root of the tree, v_0, $r(v_0) = 1$. The vertices on level 1 form a tree T_1 rooted at v_0. Denote the number of (sub)trees on level i by t_i. The vertices on level 2 will form a forest of subtrees $\{T_2^{(1)}, \cdots, T_2^{(t_2)}\}$, where the total number of subtrees is at most the sum of the number of children of all the vertices in T_1. Similarly, the vertices on level $i > 1$ form a forest of subtrees $F_i = \{T_i^{(1)}, \cdots T_i^{(t_i)}\}$. Note that all these subtrees $\{T_i^{(j)}\}$ are generated independently by the same stochastic process, as the ranks of all of the nodes in T are i.i.d. random variables. Denote $f_i = |F_i|$, and let $Y_i = \sum_{j=1}^{i} f_j$. Note that F_{i+1} can consist of at most Y_i subtrees.

We prove the following theorem.

Theorem 12. *For any infinite query tree T with bounded degree d, there exists a constant c, which depends only on d, such that for any large enough $N > 0$,*

$$Pr[|T| \geq N] \leq e^{-cN}.$$

We require the following Lemma from [14].

Lemma 17 ([14]). *Let $L \geq d+1$ be a fixed integer and let T be the d-regular infinite query tree. Then for any $1 \leq i \leq L$ and $1 \leq j \leq t_i$, there is an absolute constant c, which depends only on d, such that for all $N > 0$,*

$$\Pr[|T_i^{(j)}| \geq N] \leq e^{-cN}.$$

We first prove the following proposition:

Proposition 18. *For any infinite query tree T with bounded degree d, there exist constants μ_1 and c_1, which depend only on d, such that for any $1 \leq i \leq L-1$, and any $y_i > 0$,*

$$Pr[f_{i+1} \geq \mu_1 y_i | Y_i = y_i] \leq e^{-c_1 y_i}.$$

Proof. Fix $Y_i = y_i$. Let $\{z_1, z_2, \ldots z_{y_i}\}$ be integers such that $\forall 1 \leq i \leq y_i, z_i \geq 0$ and let $x_i = \sum_{i=1}^{y_i} z_i$. By Lemma 17, the probability that F_{i+1} consists exactly of trees of size $(z_1, z_2, \ldots z_{y_i})$ is at most $\prod_{i=1}^{y_i} e^{-cz_i} = e^{-cx_i}$. There are $\binom{x_i+y_i}{y_i}$ vectors that can realize x_i.[7] We want to bound $Pr[f_{i+1} = \mu y_i | Y_i = y_i]$ for some large enough constant $\mu > 0$. Letting $x_i = \mu y_i$, we bound it as follows:

[7] This can be thought of as y_i separators of x_i elements.

$$\begin{aligned} Pr[f_{i+1} = x_i | Y_i = y_i] &\leq \binom{x_i + y_i}{y_i} e^{-x_i} \\ &\leq \left(\frac{e \cdot (x_i + y_i)}{y_i} \right)^{y_i} e^{-cx_i} \\ &= \left(\frac{e \cdot (\mu y_i + y_i)}{y_i} \right)^{y_i} e^{-c\mu y_i} \\ &= (e \cdot (1+\mu))^{y_i} e^{-c\mu y_i} \\ &= e^{y_i(-c\mu + \ln(1+\mu)+1)} \\ &\leq e^{-c'\mu y_i}, \end{aligned}$$

for some constant $c' > 0$. It follows that

$$\begin{aligned} Pr[f_{i+1} \geq \mu y_i | Y_i = y_i] &\leq \sum_{k=\mu y_i}^{\infty} e^{-c'k} \\ &\leq e^{-c_1 y_i}, \end{aligned}$$

for some constant $c_1 > 0$. □

Proposition 18 immediately implies the following corollary.

Corollary 19. *For any infinite query tree T with bounded degree d, there exist constants μ and c, which depend only on d, such that for any $1 \leq i \leq L-1$, and any y_i,*

$$Pr[f_{i+1} \geq \mu y_i | Y_i \leq y_i] \leq e^{-cy_i}.$$

Corollary 19, which is about query trees, can be restated as follows: let $W_i = Y_i$, $q_i = y_i$ and $\sum_{i=1}^{W_i} z_i = f_{i+1}$. Furthermore, let $c'_i = c_1$ and $\mu'_i = \mu_1$ for all i. This notation yields the following proposition, which is unrelated to query trees, and which we used in Section 3:

Proposition 14 *Let W_i be a random variable. Let $z_1, z_2, \ldots z_{W_i}$ be random variables (some possibly equal to 0 with probability 1). Assume that there exist constants c and μ such that for all $1 \leq j \leq W_i$, $Pr[z_j \geq \mu N] \leq e^{-cN}$, for all $N > 0$. Then there exist constants μ_i and c'_i, which depend only on d, such that for any $q_i > 0$,*

$$Pr[\sum_{j=1}^{W_i} z_j \geq \mu_i q_i | W_i \leq q_i] \leq e^{-c'_i q_i}.$$

We need one more proposition before we can prove Theorem 12. Notice that $f_1 = |T_1|$.

Proposition 20. *For any infinite query tree T with bounded degree d, for any $1 \leq i \leq L$, there exist constants μ_i and c_i, which depend only on d, such that for and any $N > 0$,*

$$Pr[f_i \geq \mu_i N] \leq e^{-c_i N}.$$

The proof is similar to the proof of Proposition 15. We include it for completeness.

Proof. The proof is by induction on the levels $1 \leq i \leq L$, of T.

For the base of the induction, $i = 1$, by Lemma 17, we have that there exist some constants μ_1 and c_1 such that

$$Pr[f_1 \geq \mu_1 N] \leq e^{-c_1 N},$$

as $f_1 = |T_1|$.

For the inductive step, we assume that the proposition holds for levels $1, 2, \ldots, i-1$, and show that it holds for level i.

$$\begin{aligned} Pr[f_i \geq \mu_i N] =& Pr[f_i \geq \mu_i N | Y_{i-1} < \mu_{i-1} N] \cdot Pr[Y_{i-1} < \mu_{i-1} N] \\ &+ Pr[f_i \geq \mu_i N | Y_{i-1} \geq \mu_{i-1} N] \cdot Pr[Y_{i-1} \geq \mu_{i-1}] \\ \leq& Pr[f_i \geq \mu_i N | Y_{i-1} < \mu_{i-1} N] + Pr[Y_{i-1} \geq \mu_{i-1}] \\ \leq& e^{-cN} + e^{-c_{i-1} N} \qquad (3) \\ \leq& e^{-c_i N}, \end{aligned}$$

for some constant c_i. Inequality 3 stems from Corollary 19 and the inductive hypothesis. □

We are now ready to prove Theorem 12.

Proof (Proof of Theorem 12). We would like to bound $Pr[|T| = \sum_{i=1}^{L} f_i \geq \mu N]$. From Proposition 20, we have that for $1 \leq i \leq L$,

$$Pr[f_i \geq \mu_i N] \leq e^{-c_i N}.$$

A union bound on the L levels gives the required result. □

References

1. Alon, N., Babai, L., Itai, A.: A fast and simple randomized algorithm for the maximal independent set problem. Journal of Algorithms 7, 567–583 (1986)
2. Alon, N., Rubinfeld, R., Vardi, S., Xie, N.: Space-efficient local computation algorithms. In: Proc. 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1132–1139 (2012)
3. Berge, C.: Two theorems in graph theory. Proceedings of the National Academy of Sciences of the United States of America 43(9), 842–844 (1957)
4. Czygrinow, A., Hanckowiak, M.: Distributed algorithm for better approximation of the maximum matching. In: Warnow, T.J., Zhu, B. (eds.) COCOON 2003. LNCS, vol. 2697, pp. 242–251. Springer, Heidelberg (2003)
5. Edmonds, J.: Paths, trees, and flowers. Canadian Journal of Mathematics 17, 449–467 (1965)
6. Hall, P.: On representatives of subsets. J. London Math. Soc. 10(1), 26–30 (1935)
7. Harvey, N.: Algebraic structures and algorithms for matching and matroid problems. In: Proc. 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 531–542 (2006)

8. Hoepman, J.-H., Kutten, S., Lotker, Z.: Efficient distributed weighted matchings on trees. In: Flocchini, P., Gąsieniec, L. (eds.) SIROCCO 2006. LNCS, vol. 4056, pp. 115–129. Springer, Heidelberg (2006)
9. Hopcroft, J.E., Karp, R.M.: An $N^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing 2(4), 225–231 (1973)
10. Israeli, A., Itai, A.: A fast and simple randomized parallel algorithm for maximal matching. Inf. Process. Lett. 22(2), 77–80 (1986)
11. Kuhn, F., Moscibroda, T., Wattenhofer, R.: The price of being near-sighted. In: Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 980–989 (2006)
12. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2, 83–97 (1955)
13. Lotker, Z., Patt-Shamir, B., Pettie, S.: Improved distributed approximate matching. In: Proc. 20th ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 129–136 (2008)
14. Mansour, Y., Rubinstein, A., Vardi, S., Xie, N.: Converting online algorithms to local computation algorithms. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 653–664. Springer, Heidelberg (2012)
15. Mansour, Y., Vardi, S.: Local algorithmic mechanism design. Under submission elsewhere
16. Micali, S., Vazirani, V.V.: An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In: Proc. 21st Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 17–27 (1980)
17. Mucha, M., Sankowski, P.: Maximum matchings via gaussian elimination. In: FOCS, pp. 248–255 (2004)
18. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5(1), 32–38 (1957)
19. Nguyen, H.N., Onak, K.: Constant-time approximation algorithms via local improvements. In: Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 327–336 (2008)
20. Rubinfeld, R., Tamir, G., Vardi, S., Xie, N.: Fast local computation algorithms. In: Proc. 2nd Symposium on Innovations in Computer Science (ICS), pp. 223–238 (2011)

Sketching Earth-Mover Distance on Graph Metrics*

Andrew McGregor and Daniel Stubbs

University of Massachusetts Amherst
140 Governors Drive, Amherst, MA 01003
{mcgregor,dstubbs}@cs.umass.edu

Abstract. We develop linear sketches for estimating the Earth-Mover distance between two point sets, i.e., the cost of the minimum weight matching between the points according to some metric. While Euclidean distance and Edit distance are natural measures for vectors and strings respectively, Earth-Mover distance is a well-studied measure that is natural in the context of visual or metric data. Our work considers the case where the points are located at the nodes of an implicit graph and define the distance between two points as the length of the shortest path between these points. We first improve and simplify an existing result by Brody et al. [4] for the case where the graph is a cycle. We then generalize our results to arbitrary graph metrics. Our approach is to recast the problem of estimating Earth-Mover distance in terms of an ℓ_1 regression problem. The resulting linear sketches also yield space-efficient data stream algorithms in the usual way.

1 Introduction

Given two multi-sets $A, B \subseteq \mathcal{X}$ where $|A| = |B| = k$ and a metric d on $\mathcal{X}$, the *Earth-Mover Distance* (EMD) between A and B is defined as the minimum cost of a matching between A and B, i.e.,

$$\mathrm{EMD}_d(A, B) = \min_{\pi : A \to B} \sum_{a \in A} d(a, \pi(a))$$

where π ranges over all bijective mappings between A and B. Earth-Mover distance is a natural and well-studied notion of the difference between two point sets. It was initially proposed in the context of image retrieval and has been shown to correspond closely to the perceptual difference between two images [14]. While Euclidean distance and Edit distance are natural measures of dissimilarity for vectors and strings respectively, EMD is perhaps the most natural measure for metric and visual data.

Linear sketching is a popular and powerful technique for processing large data sets. See Cormode et al. [7] for a survey. The basic idea is to take random linear projections of the data set and then post-process these projections in order to

* Supported by NSF CAREER Award CCF-0953754 and associated REU Supplement.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 274–286, 2013.

evaluate properties of the original data. The main parameters of the sketch are the size, or dimension, of the projection and the time required to perform the post-processing. Important applications of sketching include processing data streams or distributed data. A significant fraction of the work on linear sketches has focused on the problem of distance estimation and estimating the Earth-Mover distance is a long-standing open question [9,12] that has remained open (in the case where the point sets lie on a $\Delta \times \Delta$ grid) despite a substantial body of work dedicated to the problem [3,4,8,10,17]. The best known results achieve a logarithmic approximation with sketches of poly-logarithmic size and an $O(1/\epsilon)$ approximation with sketches of Δ^{ϵ} size. The most relevant work to this paper is a recent paper by Brody et al. [4] in which they consider the more restricted case where $\mathcal{X}$ corresponds to the nodes of a cycle and d is the shortest-path metric on this cycle (see also Cabrelli and Molter [5] for an optimal solution in the offline, non-streaming case). In this case, they show that $(1+\epsilon)$-approximation is possible with sketches of poly-logarithmic size.

1.1 Our Techniques and Results

In this paper, we consider d to be the shortest-path metric in an arbitrary graph $G = (V, E)$ on $n = |V|$ nodes with $m = |E|$ edges. Note that the graph structure is assumed to be known in advance[1] and the input is point sets A and B of size k. Our results are as follows.

1. *Cycles:* $O(\epsilon^{-2} \operatorname{polylog} nk)$-size sketches for approximating $\mathrm{EMD}(A, B)$ up to a $(1+\epsilon)$ factor with high probability. This improves over the existing sketch of Brody et al. which used sketches of size $O(\epsilon^{-3} \operatorname{polylog} nk)$. We then show how to ensure that post-processing the sketch can be performed in $O(\operatorname{polylog} n)$ time. Our analysis also has the advantage of being significantly simpler. See Section 3.
2. *Trees:* $O(\epsilon^{-2} \operatorname{polylog} nk)$-size sketches for approximating $\mathrm{EMD}(A, B)$ up to a $(1+\epsilon)$ factor with high probability. By combining recent results on range-summable random variables by Tirthapura and Woodruff [16] with a natural path-decomposition, we show how such a sketch can be applied in the data-stream setting with $O(\operatorname{polylog} n)$ update time whereas, even in the cycle case, the existing sketch has $\Omega(n)$ update time. See Section 4.
3. *Arbitrary Graphs:* $O(\epsilon^{-2} \cdot t \cdot \operatorname{polylog} nk)$-size sketches for approximating $\mathrm{EMD}(A, B)$ up to a $(1+\epsilon)$ factor with high probability where $t = m-n+1$ is the number of edges that need to be removed from G such that the resulting graph is acyclic. This generalizes our result on cycles in which $t = 1$. While our results hold for arbitrary t, our results are most interesting in the case where there are relatively few cycles and hence t is moderate in size. See Section 5.

[1] This is in contrast to recent work in graph sketching [1,2] where the goal is to sketch the actual graph. Note that the space used in the algorithms we present will be sufficient to maintain an explicit representation of the graph structure.

Technical Approach. The general approach is follows. We define vectors $x, y \in \mathbb{R}^{|E|}$ corresponding to the two multi-sets A and B. We then relate $\mathrm{EMD}(A, B)$ to an ℓ_1-regression problem involving x, y, and a set of vectors defined by the structure of the underlying graph. To achieve our results, we first sketch the vectors, i.e., construct random projections of these vectors, and then perform the ℓ_1-regression on the sketched vectors rather than manipulating the original vectors explicitly.

2 Preliminaries

Notation. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We say an algorithm is an (ϵ, δ)-approximation for a quantity Q if the value returned $\tilde{Q}$ satisfies $\mathbb{P}\left[|Q - \tilde{Q}| < \epsilon Q\right] \geq 1 - \delta$. Given a tree $T = (V, E)$ and $u, v \in V$ we define,

$$P_T(u, v) = \{e \in E : \ e \text{ on the path between nodes } u \text{ and } v\} \ .$$

We denote the ℓ_1-norm of a vector x by $\|x\|_1 = \sum_i |x_i|$.

Sketches for ℓ_1-norm estimation. ℓ_1-norm estimation is one of the canonical sketching and data stream problems. We will make extensive use of the following result due to Kane et al. [11].

Theorem 1 (ℓ_1 Sketching [11]). *There exists a distribution ν over linear maps from $\mathbb{R}^n \to \mathbb{R}^q$ where $q = O(\epsilon^{-2} \log n \log \delta^{-1})$ and a "post-processing" function $f : \mathbb{R}^q \to \mathbb{R}$ such that for any $x \in \mathbb{R}^n$ with polynomially-bounded entries,*

$$\Pr_{M \sim \nu}[|\|x\|_1 - f(Mx)| \leq \epsilon \|x\|_1] \geq 1 - \delta \ .$$

Note that it immediately follows by rescaling δ and applying the union bound, that if we increase q to $O(\epsilon^{-2} \log n \log(t\delta^{-1}))$ we ensure that for any t vectors $\mathcal{X} = \{x_1, \ldots, x_t\}$,

$$\Pr_{M \sim \nu}[\forall x \in \mathcal{X} \ ; \ |\|x\|_1 - f(Mx)| \leq \epsilon \|x\|_1] \geq 1 - \delta \ .$$

In particular, if $\mathcal{X}$ consists of all linear combinations of some set $\{y_1, \ldots, y_r\}$ where the linear coefficients are from the set $\{-k, -k+1, \ldots, k-1, k\}$ then $t = (2k+1)^r$ and we can estimate the ℓ_1 norm of any vector $x \in \{x_1, \ldots, x_t\}$ from $O(r\epsilon^{-2} \log n \log(k\delta^{-1}))$-dimensional sketches $My_1, My_2, \ldots, My_r$ since

$$M(\sum_{i \in [r]} \lambda_i y_i) = \sum_{i \in [r]} \lambda_i M y_i \ .$$

One-Dimensional EMD. We next describe an important folklore result for sketching earth-mover distance in one dimension. For the sake of future sections, it will be helpful to describe this result in terms of graph distances when the graph is a path. Let $G = (V, E)$ be a path on n nodes, i.e., $V = \{1, 2, \ldots, n\}$ and edges

$E = \{e_1, e_2, \ldots, e_{n-1}\}$ where $e_i = \{i, i+1\}$. Suppose $A, B \subset V$ and define the distance between $i \in A$ and $j \in B$ to be shortest path distance $d(i,j) = |i-j|$.

We can relate $\mathrm{EMD}(A,B)$ to a norm estimation problem as follows. Define the vectors $x, y \in \mathbb{R}^{n-1}$ where:

$$\forall i \in [n-1]\ ; \quad x_i = |\{a \in A : i \geq a\}| \quad \text{and} \quad y_i = |\{b \in B : i \geq b\}|\ .$$

Then the following theorem establishes that $\mathrm{EMD}(A,B)$ equals $\|x-y\|_1$.

Theorem 2 (Folklore). $\mathrm{EMD}(A,B) = \|x-y\|_1$.

We will actually prove a more general result in Lemma 5 from which the above theorem follows. For intuition, suppose $A = \{i\}$ and $B = \{j\}$ and $i < j < n$. Then, $x = (0, \ldots, 0, 1, \ldots, 1)$ where the first "1" is in the i-th position and $y = (0, \ldots, 0, 1, \ldots, 1)$ where the first "1" is in the j-th position. Therefore $\|x\|_1$ and $\|y\|_1$ correspond to the distances that would be covered moving points i and j to node n. However, $y - x = (0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0)$ where $(y-x)_k = 1$ iff $i \leq k < j$ and so $\|y-x\|_1 = |j-i|$. Essentially, the effect of moving both points i and j to n cancels out along edges on which both points are being moved. The following example illustrates that the theorem applies in a less trivial case.

Example 1. Suppose $A = \{2, 3, 10\}$ and $B = \{3, 4, 8\}$ and note that $\mathrm{EMD}(A,B) = 4$. Then

$$x = (0, 1, 2, 2, 2, 2, 2, 2, 2) \quad \text{and} \quad y = (0, 0, 1, 2, 2, 2, 2, 3, 3)$$

and $\|x-y\|_1 = 4$ as required.

3 Cycles

Consider a cycle on n nodes $\{1, 2, \ldots, n\}$ and edges $e_1, e_2, \ldots, e_n$ where $e_i = \{i, i+1\}$ for $i \in [n-1]$ and $e_n = \{n, 1\}$. The basic idea for solving EMD on the cycle is to reduce it to the one-dimensional, or path metric, case by simply ignoring the last edge e_n. This has the effect of changing the distance between nodes i and j from

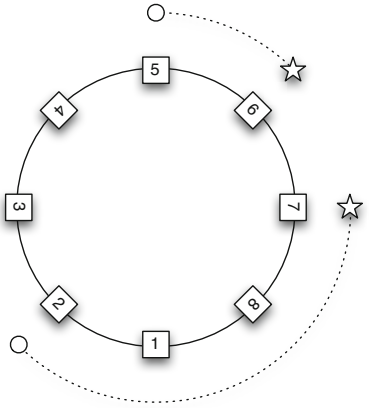
$$d(i,j) = \min(|i-j|, |i-n| + 1 + |1-j|, |i-1| + 1 + |n-j|)$$

to a new distance

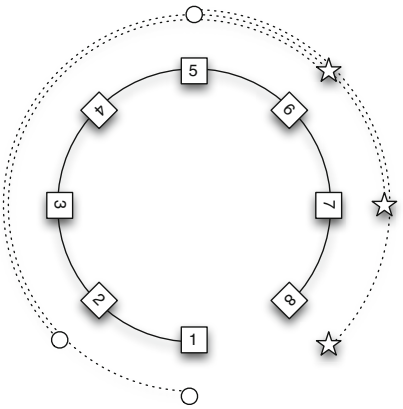
$$d'(i,j) = |i-j|\ .$$

Depending on the point sets, A and B, this can change the earth-mover distance significantly since two points that were previously close may now be far apart. For example, if $A = \{n\}$ and $B = \{1\}$ then the earth-mover distance increases from $\mathrm{EMD}_d(A,B) = 1$ to $\mathrm{EMD}_{d'}(A,B) = n-1$.

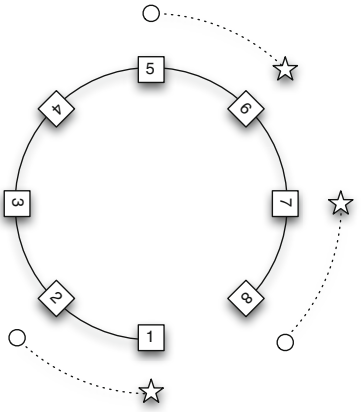
To rectify this issue, we will effectively make a series of guesses $\{-k, -k+1, \ldots, k-1, k\}$ for how many pairs of points will be paired using the edge e_n.



(a) Original Cycle Instance where $\mathrm{EMD}_d(A, B) = 4$.



(b) Linear Instance with $\lambda = 1$ where $1 + \mathrm{EMD}_{d'}(A + C_\lambda, B + C_{-\lambda}) = 14$.



(c) Linear Instance with $\lambda = -1$ where $1 + \mathrm{EMD}_{d'}(A + C_\lambda, B + C_{-\lambda}) = 4$.

Fig. 1. Reducing Cyclic EMD to Linear EMD. Points in A are denoted by circles and points in B are denoted by stars. Dotted lines indicate a minimum cost matching.

Lemma 1. *For $\lambda \in \{-k, -k+1, \ldots, k-1, k\}$, let C_λ be the multi-set consisting of λ copies of "1" if $\lambda > 0$ and $|\lambda|$ copies of "n" if $\lambda < 0$. Then,*

$$\mathrm{EMD}_d(A, B) \leq |\lambda| + \mathrm{EMD}_{d'}(A + C_\lambda, B + C_{-\lambda})$$

with equality for some $\lambda \in \{-k, -k+1, \ldots, k-1, k\}$.

Proof. Consider a bijection π between $A + C_\lambda$ and $B + C_{-\lambda}$. We first will show that π induces a bijection σ between A and B such that

$$\sum_{a \in A} d(a, \sigma(a)) \leq |\lambda| + \sum_{a \in A + C_\lambda} d'(a, \pi(a)) \ , \tag{1}$$

and this establishes the first part of the lemma.

It will be convenient to enumerate the elements of $C_\lambda = \{c_1, c_2, \ldots, c_\lambda\}$ and $C_{-\lambda} = \{d_1, d_2, \ldots, d_\lambda\}$ such that we may assume that $\pi(c_i) = d_j$ implies $i = j$. We then define σ as follows. If $\pi(a) \in B$ for $a \in A$ then define $\sigma(a) = \pi(a)$ and hence

$$d(a, \sigma(a)) = d'(a, \pi(a)) \ . \tag{2}$$

If $\pi(a) = d_i$ for some $a \in A$ and $d_i \in C_{-\lambda}$ then define $\sigma(a) = \pi(c_i)$. Hence,

$$d(a, \sigma(a)) \leq d'(a, d_i) + 1 + d'(c_i, \pi(c_i)) \ .$$

Note that there are at most $|\lambda|$ elements $a \in A$ such that $\pi(a) \in C_{-\lambda}$ and together with Eq. 2 this establishes Eq. 1.

To prove that there exists λ such that $\mathrm{EMD}_d(A, B) = |\lambda| + \mathrm{EMD}_{d'}(A + C_\lambda, B + C_{-\lambda})$ consider the bijection $\sigma = \mathrm{argmin}_\sigma \sum_{a \in A} d(a, \sigma(a))$. Suppose there are λ_1 elements $a \in A$ such that the shortest path from a to $\sigma(a)$ visits n then 1. Similarly, suppose there are λ_2 elements $a \in A$ such that the shortest path from a to $\sigma(a)$ visits 1 then n. Note that at most one of λ_1 and λ_2 is non-zero since σ is the minimal cost bijection. Then setting $\lambda = \lambda_1 - \lambda_2$ ensures $\mathrm{EMD}_d(a, \sigma(a)) = |\lambda| + \mathrm{EMD}_{d'}(A + C_\lambda, B + C_{-\lambda})$ as required.

3.1 Sketch Details

To construct the sketch we first define the vectors $x, y \in \mathbb{R}^n$ where for $i \in [n-1]$

$$x_i = |\{a \in A : i \geq a\}| \quad \text{and} \quad y_i = |\{b \in B : i \geq b\}| \ .$$

and $x_n = y_n = 0$. Define $z = x - y$ and let $c = (1, 1, \ldots, 1, 1) \in \mathbb{R}^n$.

Lemma 2. $\min_{-k \leq \lambda \leq k} \|z + \lambda c\|_1 = \mathrm{EMD}(A, B)$.

Proof. Let $z_{[n-1]}$ and $c_{[n-1]}$ be the vectors corresponding to the first $n-1$ elements of z and c respectively and note that

$$\|z + \lambda c\|_1 = |\lambda| + \|z_{[n-1]} + \lambda c_{[n-1]}\|_1 \ .$$

The proof then follows from Theorem 2 and Lemma 1.

We define the function $f(\lambda) = \|z + \lambda c\|_1$. From the above lemma, it suffices to find $\min_\lambda f(\lambda)$. From Theorem 1 (and the surrounding discussion), it is possible to compute estimates $\{\tilde{f}_\lambda\}_{\lambda \in \{-k, \ldots, k\}}$ from a $O(\epsilon^{-2} \log n \log(k\delta^{-1}))$-dimensional sketch of z such that

$$\mathbb{P}\left[\forall \lambda \in \{-k, \ldots, k\} \ : \ |\tilde{f}_\lambda - f(\lambda)| \leq \epsilon f(\lambda)\right] \geq 1 - \delta \ .$$

Hence, if we return $\min \tilde{f}_\lambda$ then we have an (ϵ, δ)-approximation for $\mathrm{EMD}(A, B)$. However, rather than evaluating every $\tilde{f}_\lambda$ to find the minimum, in the next section we next show that it is possible to find $\min_{\lambda \in \{-k, \ldots, k\}} \tilde{f}_\lambda$ while only evaluating $O(\log k)$ of the terms.

3.2 Improved Post-processing

The main observation is that since $f(\lambda) = \sum_i |z_i + \lambda c_i|$ is a sum of convex functions, $f(\lambda)$ itself is convex and can therefore be minimized by using something like a binary search.

Lemma 3. *$f(\lambda) = \|z + \lambda c\|_1$ is convex.*

However, although $f(\lambda)$ is convex, the errors in our estimates $\tilde{f}_\lambda$ of $f(\lambda)$ may violate the convexity property. To accommodate this we perform a quaternary search that includes tolerances for these errors. See Algorithm 1.

Algorithm 1. Approximate Quaternary Search

$(l, u) \leftarrow (-k, k)$
while $l \neq u$ **do**
  $(a, b, c) \leftarrow \left(\left\lfloor \frac{3l+u}{4} \right\rfloor, \left\lfloor \frac{2l+2u}{4} \right\rfloor, \left\lfloor \frac{l+3u}{4} \right\rfloor\right)$
  if $\max(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c)/\min(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c) < \frac{1+\epsilon}{1-\epsilon}$ or $\tilde{f}_b = \max(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c)$ **then**
    return $\tilde{f}_b$
  else

$$(l, u) \leftarrow \begin{cases} (a, u) & \text{if } \tilde{f}_a = \max(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c) \\ (l, c) & \text{if } \tilde{f}_c = \max(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c) \end{cases}$$

  end if
end while
return $\tilde{f}_l$

Lemma 4. *Algorithm 1 returns a value that is within a factor $1 \pm O(\epsilon)$ of $\min_\lambda f(\lambda)$.*

Proof. Let $\lambda^* = \text{argmin}_{\lambda \in \{-k,\ldots,k\}} f(\lambda)$. We first prove the invariant that l and u always satisfy $l \leq \lambda^* \leq u$. Note that it is true initially since $l = -k$ and $u = k$. Suppose it is true at a given iteration, then (by symmetry) it suffices to show that if $\max(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c)/\min(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c) \geq \frac{1+\epsilon}{1-\epsilon}$ and $\tilde{f}_a = \max(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c)$ then $a \leq \lambda^*$. Then,

$$\frac{f(a)}{\min(f(b), f(c))} \geq \frac{\tilde{f}_a/(1+\epsilon)}{\min(\tilde{f}_b/(1-\epsilon), \tilde{f}_c/(1-\epsilon))} = \frac{1-\epsilon}{1+\epsilon} \cdot \frac{\max(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c)}{\min(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c)} \geq 1 \ .$$

and hence $f(a) \geq \min(f(b), f(c))$. By the convexity of f we deduce that $a \leq \lambda^*$ as required.

It remains to show that when the algorithm terminates, the return value is sufficiently accurate.

Case 1: If $l = u$ then

$$\tilde{f}_l = (1 \pm \epsilon) f(l) = (1 \pm \epsilon) f(\lambda^*) \ .$$

Case 2: Suppose that $\max(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c)/\min(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c) < \frac{1+\epsilon}{1-\epsilon}$ and therefore

$$\frac{\max(f(a), f(b), f(c))}{\min(f(a), f(b), f(c))} < \left(\frac{1+\epsilon}{1-\epsilon}\right)^2 \ .$$

By symmetry, assume that $\lambda^* \leq b$. Then, by the convexity of f we have:

$$f(\lambda^*) \geq f(b) - 1/2 \cdot \frac{f(c) - f(b)}{1/4} = f(b)(3 - 2f(c)/f(b)) \geq (1 - O(\epsilon))\tilde{f}_b \ .$$

Case 3: Suppose that $\tilde{f}_b = \max(\tilde{f}_a, \tilde{f}_b, \tilde{f}_c)$, and assume by symmetry $\lambda^* \leq b$. Then

$$(1+\epsilon)^2 f(b) \geq (1+\epsilon)\tilde{f}_b \geq (1+\epsilon)\tilde{f}_c \geq f(c)$$

which gives us that the difference between $f(c)$ and $f(b)$ is at most $(2\epsilon + \epsilon^2)f(b)$. By convexity, the difference between $f(b)$ and $f(\lambda^*)$ is at most twice this, since λ^* is at most twice as far from b as c is, so

$$f(\lambda^*) \geq f(b) - 2(2\epsilon + \epsilon^2)f(b) = (1 - O(\epsilon))f(b) \ .$$

4 Trees

In this section, we generalize the one-dimensional case discussed in Section 2 to trees. Let $T = (V, E)$ be a tree on n nodes. Suppose $A, B \subseteq V$ where for $a \in A, b \in B$, $d(a, b)$ is the length of the unique path between a and b.

To relate EMD_d with the tree metric to ℓ_1 norms we first pick an arbitrary root r of T. Now define the vectors $x, y \in \mathbb{R}^E$ where

$$x_e = |\{a \in A : e \in P_T(a, r)\}| \quad \text{and} \quad y_e = |\{b \in B : e \in P_T(b, r)\}| \ .$$

and define $z = x - y$. Recall that $P_T(u, v)$ is the set of edges on the unique path in T between u and v. The following lemma generalizes Theorem 2 (the "root" in the path case was implicitly chosen to be node n) and will play an important role in the next section.

Lemma 5. $\|z\|_1 = \mathrm{EMD}_d(A, B)$.

Proof. For each edge $e = (u, v) \in T$ where u is a child of v, define the value

$$w_e = \big||A \cap V_u| - |B \cap V_u|\big|$$

where V_u is the set of nodes of the subtree rooted at u. Then $\mathrm{EMD}_d(A, B) = \sum_{e \in E} w_e$ since in the optimal bijection, either all points in $A \cap V_u$ will be mapped to elements in $B \cap V_u$ or vice versa and hence the edge e appears in exactly $||A \cap V_u| - |B \cap V_u||$ of the shortest paths between matched points. But $w_e = |x_e - y_e|$ since $e \in P_T(v, r)$ iff $v \in V_u$. Hence, $\mathrm{EMD}(A, B) = \sum_{e \in E} w_e = \sum_{e \in E} |x_e - y_e| = \|z\|_1$. as required.

Therefore, appealing to the ℓ_1 sketch result in Theorem 1, it immediately follows that there is an $O(\epsilon^{-2} \log n \log \delta^{-1})$-dimensional sketch that returns an (ϵ, δ) approximation for $\mathrm{EMD}_d(A, B)$ when d is a tree metric.

4.1 Improved Update Time

A naive implementation of the above algorithm requires $\Omega(n)$ update time since every update requires updating as many as $n-1$ entries of the vector. However, this can be reduced to $O(\text{polylog}\, n)$ time using the range-efficient ℓ_1 sketching algorithm of Tirthapura and Woodruff [16]. This allows contiguous segments of the vector z to be updated in $O(\text{polylog}\, n)$ time rather than $O(w\, \text{polylog}\, n)$ time where w is the length of the segment. Hence, if we can ensure that any update of z involves updating $O(\log n)$ contiguous segments we enable any update to be performed in $O(\text{polylog}\, n)$ time. To do this, we will use the following path decomposition of the tree.

Lemma 6. *For any tree $T = (V, E)$ on n nodes with ℓ leaves and root r, it is possible to decompose E into ℓ paths $P_1, \ldots, P_\ell$ such that for any $u \in V$, $P_T(u, r)$ intersects at most $O(\log \ell)$ paths.*

Proof. We define the segments $P_1, \ldots, P_\ell$ as follows. Start a segment for each leaf consisting of the edge incident on it, and associate a value of 1 with the segment. Extend these segments in the direction of the root until each reaches a node of degree ≥ 3. At each such node, we continue the segment with the highest value (ties broken arbitrarily) but add the sum of the values of the concluded segments to the value of the continued segment. Note that this value is now at least twice the value of any of the segments that were concluded. We continue in this manner until we reach the root. In the end, each edge will belong to exactly one segment. Note that the path from an arbitrary node $u \in V$ to the root can intersect with at most $\log \ell$ of the resulting segments because the value of successive intersecting segments at least doubles and the maximum value is ℓ.

Then, if we let the first $|P_1|$ elements of z correspond to P_1, the next $|P_2|$ elements correspond to P_2, etc. we ensure that when we add (or subtract) 1 to each entry corresponding to $P_T(u, r)$ for some u, this involves only $O(\log n)$ updates of contiguous intervals.

5 Arbitrary Graphs

In this final section, we generalize all our previous results and design a sketch for earth-mover distance for arbitrary graph metrics. Let $G = (V, E)$ be a graph on n nodes. Define a metric d where for $a, b \in V$, $d(a, b)$ is the length of the shortest path between a and b in G.

The approach to estimating $\text{EMD}_d(A, B)$ is to reduce to the tree-metric case solved in the previous section. This naturally extends the approach in Section 3 where we reduced the cycle case to the path-metric case. Specifically, let $T = (V, E_T)$ be an arbitrary spanning tree and let $F = E \setminus E_T$. For example, see Figure 2 where $E_T = \{e_1, e_2, e_3, e_4\}$ and $F = \{f_1, f_2\}$. The tree T defines a metric d' where for $a, b \in V$, $d'(a, b)$ is the length of the shortest path between a and b in T.

The next lemma shows that it is possible to express $\mathrm{EMD}_d(A, B)$ in terms of $\mathrm{EMD}_{d'}(A', B')$ where $A \subseteq A'$ and $B \subseteq B'$. The lemma is a generalization of Lemma 1.

Lemma 7. *For $f = (u, v) \in F$ and $\lambda \in \{-k, -k+1, \ldots, k-1, k\}$, let C^f_λ be the multi-set consisting of λ copies of "u" if $\lambda > 0$ and $|\lambda|$ copies of "v" if $\lambda < 0$. Then,*

$$\mathrm{EMD}_d(A, B) \leq \sum_{f \in F} |\lambda_f| + \mathrm{EMD}_{d'}(A + \sum_{f \in F} C^f_{\lambda_f}, B + \sum_{f \in F} C^f_{-\lambda_f}) \tag{3}$$

with equality for some set of coefficients λ_f.

Proof. Consider a bijection π between $A' = A + \sum_{f \in F} C^f_{\lambda_f}$ and $B' = B + \sum_{f \in F} C^f_{-\lambda_f}$. We will show that π induces a bijection σ between A and B such that

$$\sum_{a \in A} d(a, \sigma(a)) \leq \sum_{f \in F} |\lambda_f| + \sum_{a \in A'} d'(a, \pi(a)) \ , \tag{4}$$

and this will establish the first part of the lemma.

It will be convenient to enumerate the elements of $C^f_{\lambda_f}$ and $C^f_{-\lambda_f}$:

$$C^f_{\lambda_f} = \{c^f_1, c^f_2, \ldots\} \quad \text{and} \quad C^f_{-\lambda_f} = \{d^f_1, d^f_2, \ldots\}$$

such that we may assume that $\pi(c^f_i) = d^f_j$ implies $i = j$. If $\pi(a) \in B$, let $\sigma(a) = \pi(a)$. Otherwise, define the sequence:

$$s_a = (a, d^{f_1}_{i_1}, c^{f_1}_{i_1}, d^{f_2}_{i_2} \ldots, c^{f_{k-1}}_{i_{k-1}}, d^{f_k}_{i_k}, c^{f_k}_{i_k}, b)$$

where each successive element is uniquely defined by π and the indexing of the elements in each $C^f_{\lambda_f}$ and $C^f_{-\lambda_f}$:

$$d^{f_1}_{i_1} = \pi(a) \ , \quad d^{f_2}_{i_2} = \pi(c^{f_1}_{i_1}) \ , \quad \ldots \quad , \quad d^{f_k}_{i_k} = \pi(c^{f_{k-1}}_{i_{k-1}}) \ , \quad \text{and} \quad b = \pi(c^{f_k}_{i_k}) \ .$$

Given s_a, define $\sigma(a) = b$, i.e., we match a with the last element of the sequence. Note that

$$d(a, \pi(a)) \leq d'(a, d^{f_1}_{i_1}) + 1 + d'(c^{f_1}_{i_1}, d^{f_2}_{i_2}) + 1 + \ldots + 1 + d'(c^{f_k}_{i_k}, b) \ .$$

Summing over all $a \in A$, gives Eq. 4 since each pair (d^f_i, c^f_i) appears in at most one sequence because π is a bijection.

To prove that there exists a set of coefficients such that Eq. 3 is tight, consider the bijection $\sigma = \mathrm{argmin}_\sigma \sum_{a \in A} d(a, \sigma(a))$. Then, for each $f = (u, v)$, let

$$\begin{aligned} \lambda_f = &|\{a \in A : \ u \text{ appears before } v \text{ on path between } a \text{ and } \sigma(a)\}| \\ &-|\{a \in A : \ u \text{ appears before } v \text{ on path between } \sigma(a) \text{ and } a\}| \ . \end{aligned}$$

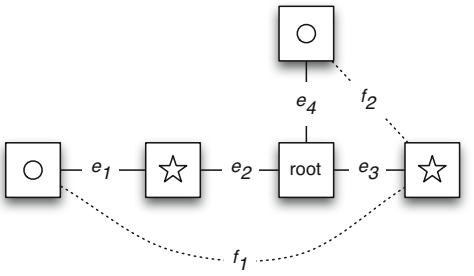


Fig. 2. An instance of earth-mover distance on an arbitrary graph metric. See the text in Example 2. Points in A are denoted by circles and points in B are denoted by stars.

Then,

$$\mathrm{EMD}_{d'}(A + C^f_{\lambda_f}, B + C^f_{-\lambda_f}) \leq \mathrm{EMD}_d(A, B) - \sum_{f \in F} |\lambda_f|$$

since with the addition of the $C^f_{\lambda_f}$ and $C^f_{-\lambda_f}$ sets we can consider the matching between $A + C^f_{\lambda_f}$ and $B + C^f_{-\lambda_f}$ induced by removing all edges $f \in F$.

5.1 Sketch Details

For a graph $G = (V, E)$, let $T = (V, E_T)$ be an arbitrary spanning tree with root r. Define the vectors $x, y \in \mathbb{R}^E$ and $z = x - y$ where

$$x_e = \begin{cases} |\{a \in A : e \in P_T(a, r)\}| & \text{if } e \in E_T \\ 0 & \text{otherwise} \end{cases}$$

$$y_e = \begin{cases} |\{b \in B : e \in P_T(b, r)\}| & \text{if } e \in E_T \\ 0 & \text{otherwise} \end{cases} .$$

For each $f = (u, v) \in F$, we define a vector c^f where

$$c^f_e = \begin{cases} 1 & \text{if } e \in P_T(u, r) \setminus P_T(v, r) \\ -1 & \text{if } e \in P_T(v, r) \setminus P_T(u, r) \\ 1 & \text{if } e = f \\ 0 & \text{otherwise} \end{cases}$$

The intuition behind the definition of c^f is that if z corresponds to point sets A and B, then $z + \lambda_f c^f$ corresponds to point sets $A + C^f_{\lambda_f}$ and $B + C^f_{-\lambda_f}$.

Example 2. Consider the instance in Figure 2. In this case

$$x = (1, 1, 0, 1, 0, 0) \quad , \quad y = (0, 1, 1, 0, 0, 0) \quad , \quad z = (1, 0, -1, 1, 0, 0)$$

$$c^{f_1} = (1, 1, -1, 0, 1, 0) \quad \text{and} \quad c^{f_2} = (0, 0, 1, -1, 0, 1) \ .$$

Note that $\|z+0c^{f_1}+1c^{f_2}\|_1 = \|(1,0,0,0,0,1)\|_1 = \mathrm{EMD}_d(A,B)$ and for arbitrary λ_1, λ_2 we have

$$\|z + \lambda_1 c^{f_1} + \lambda_2 c^{f_2}\|_1 \geq \mathrm{EMD}(A,B) \ .$$

Lemma 8. $\min_{-k\leq\lambda_1,\ldots,\lambda_t\leq k} \|z + \sum_{f\in F} \lambda_f c^f\|_1 = \mathrm{EMD}_d(A,B)$.

Proof. Let $z_{[n-1]}$ and $c^f_{[n-1]}$ be the vectors corresponding to the first $n-1$ elements of z and c^f for each f. Note that

$$\|z + \sum_{f\in F} \lambda_f c^f\|_1 = \sum_{f\in F} |\lambda_f| + \|z_{[n-1]} + \sum_{f\in F} \lambda_f c^f_{[n-1]}\|_1 \ .$$

The proof then follows from Lemma 7 and Theorem 2.

Extending the idea in Section 3, we now define the function $f(\lambda_1,\ldots,\lambda_t) = \|z + \sum_{f\in F} \lambda_f c^f\|_1$. From the above lemma, it suffices to estimate

$$\min_{-k\leq\lambda_1,\ldots,\lambda_t\leq k} f(\lambda_1,\ldots,\lambda_t) \ .$$

From Theorem 1 (and the surrounding discussion), it is possible to compute estimates $\{\tilde{f}_{\lambda_1,\ldots,\lambda_t}\}_{-k\leq\lambda_1,\ldots,\lambda_t\leq k}$ from a $O(t\epsilon^{-2}\log n\log(k\delta^{-1}))$-dimensional sketch of z such that with probability at least $1-\delta$, for all $-k \leq \lambda_1,\ldots,\lambda_t \leq k$

$$|f(\lambda_1,\ldots,\lambda_t) - \tilde{f}_{\lambda_1,\ldots,\lambda_t}| \leq \epsilon f(\lambda_1,\ldots,\lambda_t) \ .$$

Hence, if we return the minimum estimate then we have an (ϵ,δ) approximation for $\mathrm{EMD}(A,B)$. However, as in the cycle case, rather than evaluating every $\tilde{f}_{\lambda_1,\ldots,\lambda_t}$ to find the minimum, it is possible to find the minimum more efficiently. One option is to exploit the convexity of f as in Section 3 using a recursive regression algorithm [13] or to use recent results on robust regression via subspace embeddings [6,15].

References

1. Ahn, K.J., Guha, S., McGregor, A.: Analyzing graph structure via linear measurements. In: SODA, pp. 459–467 (2012)
2. Ahn, K.J., Guha, S., McGregor, A.: Graph sketches: sparsification, spanners, and subgraphs. In: PODS, pp. 5–14 (2012)
3. Andoni, A., Ba, K.D., Indyk, P., Woodruff, D.P.: Efficient sketches for earth-mover distance, with applications. In: FOCS, pp. 324–330 (2009)
4. Brody, J., Liang, H., Sun, X.: Space-efficient approximation scheme for circular earth mover distance. In: Fernández-Baca, D. (ed.) LATIN 2012. LNCS, vol. 7256, pp. 97–108. Springer, Heidelberg (2012)
5. Cabrelli, C., Molter, U.: A linear time algorithm for a matching problem on the circle. Inf. Process. Lett. 66(3), 161–164 (1998)
6. Clarkson, K.L., Drineas, P., Magdon-Ismail, M., Mahoney, M.W., Meng, X., Woodruff, D.P.: The fast cauchy transform and faster robust linear regression. In: SODA, pp. 466–477 (2013)

7. Cormode, G., Garofalakis, M.N., Haas, P.J., Jermaine, C.: Synopses for massive data: Samples, histograms, wavelets, sketches. Foundations and Trends in Databases 4(1-3), 1–294 (2012)
8. Indyk, P.: A near linear time constant factor approximation for euclidean bichromatic matching (cost). In: SODA, pp. 39–42 (2007)
9. Indyk, P., McGregor, A., Newman, I., Onak, K. (eds.): Open Problems in Data Streams, Property Testing, and Related Topics (2011), http://www.cs.umass.edu/~mcgregor/papers/11-openproblems.pdf
10. Indyk, P., Price, E.: K-median clustering, model-based compressive sensing, and sparse recovery for earth mover distance. In: STOC, pp. 627–636 (2011)
11. Kane, D.M., Nelson, J., Porat, E., Woodruff, D.P.: Fast moment estimation in data streams in optimal space. In: STOC, pp. 745–754 (2011)
12. McGregor, A. (ed.): Open Problems in Data Streams and Related Topics (2007), http://www.cse.iitk.ac.in/users/sganguly/data-stream-probs.pdf
13. McGregor, A., Rudra, A., Uurtamo, S.: Polynomial fitting of data streams with applications to codeword testing. In: STACS, pp. 428–439 (2011)
14. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision 40(2), 99–121 (2000)
15. Sohler, C., Woodruff, D.P.: Subspace embeddings for the l_1-norm with applications. In: STOC, pp. 755–764 (2011)
16. Tirthapura, S., Woodruff, D.P.: Rectangle-efficient aggregation in spatial data streams. In: PODS, pp. 283–294 (2012)
17. Verbin, E., Zhang, Q.: *Rademacher-sketch*: A dimensionality-reducing embedding for sum-product norms, with an application to earth-mover distance. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 834–845. Springer, Heidelberg (2012)

Online Multidimensional Load Balancing

Adam Meyerson[1,⋆], Alan Roytman[2,⋆⋆], and Brian Tagiku[1,⋆⋆⋆]

[1] Google, Inc.
{awmeyerson,btagiku}@google.com
[2] Computer Science Department
University of California, Los Angeles
alanr@cs.ucla.edu

Abstract. Energy efficient algorithms are becoming critically important, as huge data centers and server farms have increasing impact on monetary and environmental costs. Motivated by such issues, we study online load balancing from an energy perspective. Our framework extends recent work by Khuller, Li, and Saha (SODA 2010) to the online model. We are given m machines, each with some energy activation cost c_i and d dimensions (i.e., components). There are n jobs which arrive online and must be assigned to machines. Each job induces a load on its assigned machine along each dimension. We must select machines to activate so that the total activation cost of the machines falls within a budget B and the largest load over all machines and dimensions (i.e., the makespan) by assigning jobs to active machines is at most Λ.

We first study the model in which machines are unrelated and can have arbitrary activation cost. In this problem, which we call Machine Activation, we extend previous work to handle jobs which arrive online. We consider a variant where the target makespan Λ and budget B are given. The first main result is an online algorithm which is $O(\log(md)\log(nm))$-competitive on the load Λ and $O(d\log^2(nm))$-competitive on the energy budget B. We also address cases where one parameter is given and we are asked to minimize the other, or where we want to minimize a convex combination of the two. Running our previous algorithm in phases gives results for these variants. We prove lower bounds indicating that the effect on the competitive ratio due to multiple phases is necessary.

Our second main result is in the same setting except all machines are identical and have no activation cost. We call this problem Vector Load Balancing, our objective is to minimize the largest load induced over all machines and dimensions (makespan) and the sum of the largest induced load on each machine (energy). We give an online algorithm that is $O(\log d)$-competitive on makespan, which improves even on the best prior offline result, and $O(\log d)$-competitive on the sum of induced loads if the target makespan is given; without this knowledge we show that it is impossible to get a competitive ratio independent of m.

⋆ This work was done while the author was at the University of California, Los Angeles.

⋆⋆ Research partially supported by NSF grants IIS-1065276, CCF-1016540, CNS-1118126, and CNS-1136174.

⋆⋆⋆ This work was done while the author was at the University of California, Los Angeles.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 287–302, 2013.

1 Introduction

1.1 Motivation and Applications

Billions of dollars are spent every year to power computer systems, and any improvement in power efficiency could lead to significant savings [25]. With the rise of huge data centers and server farms, energy costs and cooling costs have become a significant expense as demand for computing power, servers, and storage grows. Indeed, energy costs and cooling costs are likely to exceed the cost of acquiring new hardware and servers. Managers of data centers wish to optimize power consumption without sacrificing any performance to minimize energy costs and cooling costs due to heat dissipation [22]. For these reasons, algorithms for energy efficient scheduling are very valuable, and even a small improvement could lead to significant savings and a positive impact on the environment.

As data centers and server farms grow in size, it becomes increasingly important to decide which machines should stay active and which should be shut down. In particular, there are opportunities for energy conservation and monetary savings due to cooling costs [22]. Moreover, these larger data centers have huge fluctuations in work loads, ranging from very high peaks to very low valleys. Hence, when demand is low, it is possible to shut down some machines which would allow for significant savings [17]. Certain jobs may need access to particular machines (due to data availability or device capability), and hence it makes sense to consider a subset of machines to activate for a given set of jobs. Once the appropriate machines have been activated, the pending jobs may then be scheduled. Note that the process of choosing which machines to activate is naturally an online problem, since jobs arrive dynamically over time.

In [24, 25, 27], methods were developed to measure power consumption at a high sampling rate. This allows us to measure energy requirements of recurring jobs at various speeds, and also to use machine learning techniques to estimate these requirements for new jobs. In [25] measurements were made for the energy effects of resource contention between jobs. The results indicated that jobs which make heavy use of different system components can be parallelized in a power-efficient manner, whereas jobs which make heavy use of the same components do not parallelize well. Their results motivate our belief that optimizing the scheduling of jobs to minimize power consumption is a non-trivial problem in real systems. From the standpoint of virtualization software such as VMware, it is important to effectively distribute jobs among machines so that resource contention (and thus energy use) is minimized. Multidimensional load balancing has applications in cutting stock, resource allocation, and implementation of databases for share-nothing environments [10–12]. This problem also has applications in multidimensional resource scheduling for parallel query optimization in databases. Query execution typically involves multidimensionality, particularly among time-sharing system resources such as the CPU or disk [11]. This motivates representing jobs as having d dimensions where each dimension represents the load induced by the job on the component. Real jobs often involve more than two components (and real network speeds depend on internet congestion). It is important to model the load

placed on various system components (processor, network, memory, etc.) and the key to obtaining good performance (both in terms of completion times and power) is to balance the loads appropriately.

1.2 Problem Definitions

We consider two problems in this domain. Our first problem considers online allocation of jobs to unrelated machines with arbitrary activation costs and d dimensions, which we call the Machine Activation problem:

Definition 1 (Machine Activation Problem). *We are given a set of m unrelated machines each with d dimensions (i.e., components such as CPU, memory, network) and an activation cost c_i. Moreover, a set of n jobs j arrive online, each inducing a load of p_{ij}^k if assigned to machine i on dimension k. We must select a set A of machines to activate such that $\sum_{i \in A} c_i \leq B$ for a constraint budget B, and assign jobs to active machines such that the total p_{ij}^k for jobs assigned to machine i along dimension k is at most a load constraint Λ.*

The main setting we study is when Λ and B are given to the algorithm. Our competitive guarantees hold if there is an offline integral solution with makespan Λ and budget B. Note that we also consider variants in which the load Λ or budget B may not necessarily be specified, in which case we seek to minimize the corresponding objective.

Our second problem considers online allocation of jobs to identical machines with multiple components and no activation costs.

Definition 2 (Vector Load Balancing Problem). *We have m identical machines each with d components. Jobs $\boldsymbol{p}_j$ arrive online and are to be assigned to machines upon arrival. Here, the k'th coordinate p_j^k gives the load placed on component k by job j. Let ℓ_i^k denote the sum of p_j^k over all jobs j on machine i. The* load *ℓ_i of machine i is $\max_k \ell_i^k$. Our goal is to simultaneously minimize the makespan, $\max_i \ell_i$, and the energy, $\sum_i \ell_i$.*

1.3 Our Contributions and Techniques

For the Machine Activation problem, we design an online algorithm in Section 2 that is $O(\log(md)\log(nm))$-competitive on the load and $O(d\log^2(nm))$-competitive on the energy budget for the case when the load Λ and the budget B are given. It extends the result of [17] to an online setting and the work of [4] to the multidimensional setting. Our main technique considers the linear relaxation of an integer program. We approximately solve the linear relaxation online as constraints arrive, such that everything except the budget and load constraints is feasible. We develop a novel algorithm and technique for analyzing the primal linear program, with a unique combination of multiplicative and additive updates. This innovative approach ensures that key inequalities are feasible and keeps the total number of iterations of the algorithm small. Our fractional solution

is $O(\log(md)\log(nm))$-competitive on the load and $O(d\log(nm))$-competitive on the energy budget. Our analysis includes a non-trivial potential function to obtain our competitive result on the load. We then describe an online randomized rounding scheme to produce a competitive ratio of $O(\log(md)\log(nm))$ on the load and $O(d\log^2(nm))$ on the budget. Combining our approach with the rounding scheme of the Generalized Assignment Problem [26] also gives an offline result similar to [17] with a substantially simpler rounding scheme. Since our problem generalizes both set cover even in the one-dimensional setting (by setting the processing times to 0 or ∞) and load balancing (by setting all c_i to 0), we can apply lower bounds from the online versions of these problems from [1, 2] to get polylogarithmic online lower bounds. We also show that no deterministic algorithm can be competitive.

In Section 3, we give upper and lower bounds for variants of the Machine Activation problem. We consider variants where one parameter (either B or Λ) is given up front and the goal is to minimize the other. We obtain our positive results by running our online algorithm from Section 2 in phases. Though this induces a logarithmic dependence on the value of the optimal solution, we show that such dependence is necessary for a fully online algorithm, suggesting a semi-online algorithm that is given a good estimate of the optimum performs much better. Lastly, we consider the case where neither parameter is given and the goal is to minimize a linear combination of the maximum load and the energy cost (say $c_\Lambda \Lambda^* + c_B B^*$, where Λ^*, B^* are the makespan and energy cost of a schedule, respectively). Our algorithm is $O(d\log^2(nm))$-competitive on this objective.

In Section 4, for the Vector Load Balancing problem, we design an online algorithm which is $O(\log d)$-competitive on the makespan, improving even the best known offline result [9]. Our algorithm is simultaneously $O(\log d)$-competitive on the energy usage if we are given a small piece of information (the maximum load induced by any single job on any single component). Without this information, we show our competitive ratios on the two criteria must have product $\Omega(\min(d, \log m))$. Our main technique involves adapting a result of Aspnes et al. [2] which works by assigning jobs greedily based on an exponential cost function. A direct application of their proof technique requires a near-optimum offline solution (which we do not have) and obtains competitive ratio $O(\log md)$, matching random assignment [9]. We modify their technique to make use of suitable rough bounds on the optimum load and exploit the fact that our machines are identical to obtain an $O(\log d)$ bound. Our analysis also includes the use of a non-trivial potential function argument. The work of [5] considers a similar problem from a bin packing perspective. They also develop a greedy algorithm based on exponential cost functions and some of the techniques they use are similar, though the works were done independently of one another. It is perhaps interesting that similar algorithms can be used for both online multidimensional bin packing and load balancing to obtain strong competitive ratios for the two problems.

1.4 Related Work

For the Machine Activation problem, the recent work of [17] studies the same problem in the offline setting. They give an algorithm for the unrelated Machine Activation problem which produces a schedule with makespan at most $(2+\varepsilon)\Lambda$ and activation cost at most $2(1+\frac{1}{\varepsilon})(\ln \frac{n}{OPT}+1)B$ for any $\varepsilon > 0$ (where OPT is the number of active machines in the optimal solution), assuming there is a schedule with makespan Λ and activation cost B. They also give a polynomial time approximation scheme for the uniformly related parallel machines case (where machine i has speed s_i and $p_{ij} = \frac{p_j}{s_i}$), which outputs a schedule with activation cost at most B and makespan at most $(1+\varepsilon)\Lambda$, for any $\varepsilon > 0$. In [20], a generalized version of the Machine Activation problem is considered in the offline setting where each machine's activation cost is a function of the load assigned to the machine. An algorithm is given which assigns at least $n - \epsilon$ jobs fractionally with cost at most $(1 + \ln(n/\epsilon))OPT$.

In addition, [20] studies an offline version of the Machine Activation problem in which each machine has d linear constraints. For this version, they give a solution which is $O(\frac{1}{\epsilon}\log n)$ times the optimal activation cost while breaking the d machine constraints by at most a factor of $2d + \epsilon$. This can be compared with our online, multidimensional guarantees of $O(\log(nm)\log(md))$ on the load and $O(d\log^2(nm))$ on the activation cost. There is also the recent work of [4], which we extend to the multidimensional setting. The work of [4] studies several problems. For their generalized framework, which is referred to as the Online Mixed Packing and Covering (OMPC) problem, a deterministic $O(\log P \log(v\rho\kappa))$-competitive algorithm is given, where P is the number of packing constraints, v is the maximum number of variables in any constraint, and ρ (respectively, κ) is the ratio of the maximum to the minimum non-zero packing (respectively, covering) coefficient respectively. Hence, if all coefficients are either 0 or 1, this is $O(\log P \log v)$-competitive. Note that we cannot simply apply the general scheme in [4] for OMPC, since our packing constraints are not given offline (indeed, this is precisely where the online nature of our problem comes into play). The OMPC framework only models programs in which packing constraints are given offline. The work of [4] also studies a problem called Unrelated Machine Scheduling with Startup Costs (UMSC), which is similar to our problem in the single dimensional case. In particular, when $d = 1$, they give an $O(\log m)$-competitive result on the makespan and an $O(\log(mn)\log m)$-competitive result on the energy budget. We extend this result to the multidimensional setting. Note that it is not clear how to adapt their algorithm to the multidimensional setting, and we develop our own framework which uses a novel combination of additive and multiplicative updates in our fractional algorithm.

In [2], they consider the online load balancing problem without activation costs. They give an $O(\log m)$-competitive algorithm for unrelated machines and an 8-competitive algorithm for related machines. In [3], they consider the online load balancing problem without activation costs where the load on a machine is measured according to the L_p norm. Their main result is that the greedy algorithm is $O(p)$-competitive under the L_p norm, and any deterministic algorithm

must be $\Omega(p)$-competitive. In [26], they give the first constant approximation (offline) for the unrelated machines case. In [8], the identical machines case is studied in the online setting without activation costs. It is shown that greedy is globally $O(\log m)$-balanced in the restricted assignment model and globally $O(\log m)$-fair in the 1-∞ model (see [8] for details). Our problem and techniques are substantially different, as we do not design or analyze a greedy algorithm. For a survey on power management and energy minimization, see [15]. Also, see [23] for a comprehensive survey on online scheduling.

For the Vector Load Balancing problem, the single-dimensional case has an offline PTAS [14] and a $(2-\varepsilon)$-competitive online algorithm for a small fixed ε [7]. The multidimensional version was introduced by [9], which includes an offline PTAS with running time exponential in the number of dimensions d and an offline $O(\log^2 d)$-approximation with polynomial running time. They also prove that a simple randomized algorithm is $O(\frac{\log md}{\log\log md})$-competitive (with m machines and d dimensions), and that no polynomial-time offline algorithm can attain a constant-factor approximation under standard complexity assumptions.

Most of the prior work (including a substantial portion of [9]) focuses on vector bin packing where we have a *hard constraint* on the makespan and must minimize the number of machines (bins). Some of these results include [6, 9, 13, 16, 18, 19, 21] with the best being $O(\log d)$-approximations. The recent work of [5] studies vector bin packing in the online setting. The algorithm they design for their positive result is similar to ours for Vector Load Balancing, though the two works were done independently of each other.

2 Machine Activation

2.1 LP and Algorithm

We formulate the problem as an integer program, where $y_i = 1$ means machine i is activated, and $x_{ij} = 1$ means job j is assigned to machine i. We assume the target load Λ and budget B are given, and that there is an offline integral solution with makespan at most Λ and budget at most B.

1. For all $1 \le i \le m$, we have $0 \le y_i \le 1$.
2. For all $1 \le i \le m$ and $1 \le j \le n$ we have $x_{ij} \ge 0$.
3. For all j, we have $\sum_{i=1}^{m} x_{ij} \ge 1$.
4. For all i, j we have $x_{ij} \le y_i$.
5. For all i, k, we have $\sum_{j=1}^{n} x_{ij} p_{ij}^k \le \Lambda y_i$.
6. We have $\sum_{i=1}^{m} y_i c_i \le B$.

Our goal is to design an online algorithm to solve the integer version of this linear system, while violating constraints 5 and 6 by at most a bounded factor. We will do this by first providing an online solution to the linear relaxation above, which may also violate the first constraint by possibly having $y_i \ge 1$, then describe an online rounding technique to produce an integer solution in Section 2.3. We will assume that either $\frac{B}{m} \le c_i \le B$ or $c_i = 0$ for all i (if more then discard machine i as offline cannot use it, if less then simply buy machine

i and assume $c_i = 0$ for a constant factor increase in the total cost). Note that we can normalize B to anything we wish - we will choose $B = \Theta(m)$ so that any non-zero c_i is at least a constant (for instance, at least 1). We define $q_{ij}^k = p_{ij}^k / \Lambda$ and $\ell_i^k = \sum_j q_{ij}^k \max\{x_{ij} - \frac{1}{jm}, 0\}$. Let $a \geq 1$ be a constant to be set later.

We first give some intuition for our algorithm. When a job j arrives, most constraints are satisfied except for $\sum_{i=1}^m x_{ij} \geq 1$. To fix this, we need to raise the x_{ij} variables until this inequality is satisfied. However, this may cause other inequalities such as $x_{ij} \leq y_i$ and $\sum_{j=1}^n x_{ij} p_{ij}^k \leq \Lambda y_i$ to be violated. Hence, we will only increase x_{ij} if $x_{ij} \leq y_i$ will continue to hold (if we do raise x_{ij}, then we also increase y_i to satisfy the load inequalities). If increasing x_{ij} would cause $x_{ij} > y_i$, then we simply increase y_i. We increase x_{ij} multiplicatively, so that the larger x_{ij} is, the larger the increase. Moreover, it seems intuitively clear that we should increase x_{ij} for job j less aggressively if p_{ij}^k, c_i, or ℓ_i^k is large (in fact, we penalize machine i with an exponential cost function for the load ℓ_i^k to obtain our competitive ratio on the load constraints). We also increase y_i multiplicatively whenever it is too small (again, intuitively, y_i should be increased less aggressively if c_i is large). When a variable is small, it may take many iterations for multiplicative updates to increase its value substantially. We use additive updates to avoid this issue, and couple the additive updates with multiplicative ones to achieve our feasibility and competitive guarantees by keeping the number of iterations in our algorithm small. See Algorithm 1 for details.

1 Initialize $x_{ij} \leftarrow 0$ for all i, j and $y_i \leftarrow 0$ for all i
2 When job j arrives, set $x_{ij} \leftarrow \frac{1}{jm}$ for each i such that $p_{ij}^k \leq \Lambda$ for all k and $y_i \leftarrow y_i + \frac{1}{jm}$ for all i
3 **while** job j has $\sum_{i=1}^m x_{ij} < 1$ **do**
4 **for** each $1 \leq i \leq m : p_{ij}^k \leq \Lambda$ for all k **do**
5 Set $z_{ij} \leftarrow \frac{x_{ij}}{\sum_k p_{ij}^k (a^{\ell_i^k} + c_i) + 1}$
6 **if** $x_{ij} + z_{ij} \leq y_i$ **then**
7 Set $x_{ij} \leftarrow \min\{x_{ij} + z_{ij}, 1\}$ and $y_i \leftarrow y_i + z_{ij} \max_k q_{ij}^k$
8 **else**
9 Set $y_i \leftarrow y_i(1 + \frac{1}{d\Lambda c_i})$

Algorithm 1. Fractional Assignment

2.2 Analysis

We prove certain feasibility properties which our algorithm maintains.

Theorem 1. *Inequalities* 3, 4, *and* 5 *of the linear program are satisfied.*

Proof. Inequality 3 follows from the termination condition of step three of the algorithm. The other two inequalities hold initially since all variables are zero.

When j arrives, we set $x_{ij} \leftarrow \frac{1}{jm}$ for every i where $q_{ij}^k \leq 1$ along each dimension k, but we also increase all y_i by $\frac{1}{jm}$, so both inequalities will continue to hold.

Later, we might increase x_{ij} by some z_{ij}. However, we will only do this if $x_{ij} + z_{ij} \leq y_i$, so $x_{ij} \leq y_i$ still holds. When we increase x_{ij} in this way, we will also increase y_i by $z_{ij} \max_k q_{ij}^k$, which guarantees that $\sum_j x_{ij} p_{ij}^k \leq \Lambda y_i$ will continue to hold for all i and all k. □

Each time through the loop at line three of the algorithm will be called a **reinforcement** step. Let r_j represent the number of reinforcement steps which occur on the arrival of j.

Lemma 1. *For $a \geq 1$, when job j arrives, the r_j reinforcement steps increase $\sum_{i=1}^m \sum_{k=1}^d a^{\ell_i^k}$ by at most $\frac{a-1}{\Lambda} r_j$.*

Proof. Each reinforcement step increases ℓ_i^k by at most $z_{ij} q_{ij}^k$ for each i and k. Thus the total increase in the summation is bounded by $\sum_{i=1}^m \sum_{i=1}^k a^{\ell_i^k} (a^{z_{ij} q_{ij}^k} - 1)$. By the definition of z_{ij}, we will always have $z_{ij} q_{ij}^k \leq 1$. In general for any $a \geq 1$ we will have $a^x - 1 \leq (a-1)x$ whenever $0 \leq x \leq 1$, and applying this allows us to bound the summation by $\sum_{i=1}^m \sum_{k=1}^d a^{\ell_i^k} (a-1)(z_{ij} q_{ij}^k)$. We can substitute $z_{ij} \leq \frac{x_{ij}}{\sum_k p_{ij}^k a^{\ell_i^k} + 1}$ and use the fact that $\sum_i x_{ij} < 1$ prior to the reinforcement to get that $\sum_{i=1}^m \frac{a-1}{\Lambda} x_{ij} \leq \frac{a-1}{\Lambda}$. □

Lemma 2. *The total number of reinforcement steps is bounded by $\sum_j r_j \leq \Lambda (\log nm)(\sum_i \sum_k a^{\ell_i^k} + 2dB) + n(1 + \log mn)$.*

Proof. We split the reinforcements occurring on the arrival of j into two sets. Suppose i is the machine to which j is assigned by the optimum solution. We have $\bar{r}_j$ **load reinforcing** steps where $x_{ij} + z_{ij} \leq y_i$ and $\hat{r}_j$ **cost reinforcing** steps where the opposite was true. Clearly $r_j = \bar{r}_j + \hat{r}_j$.

Each time a load reinforcing step occurs, x_{ij} increases by a factor of at least $1 + \frac{1}{\sum_k p_{ij}^k (a^{\ell_i^k} + c_i) + 1}$ (except for possibly the last and only reinforcement step for job j in which x_{ij} is set to 1 instead of $x_{ij} + z_{ij}$ at line seven). Thus every $\sum_k p_{ij}^k (a^{\ell_i^k} + c_i) + 1$ such steps increase x_{ij} by a constant factor. Since x_{ij} is initially at least $\frac{1}{mn}$, the total number of such steps is bounded by $\log(nm)(\sum_k p_{ij}^k (a^{\ell_i^k} + c_i) + 1) + 1$. Summing over all machines i used by the optimum solution, we get $\sum_j \bar{r}_j \leq \Lambda \log(nm)(\sum_{i \in OPT} \sum_k a^{\ell_i^k} + dB) + n \log(mn) + n$.

Consider all cost reinforcing steps for jobs which the optimum assigns to machine i. The initial value of y_i is $\frac{1}{m}$. Each time we apply a cost reinforcing step, we increase this by a multiplicative $1 + \frac{1}{d\Lambda c_i}$. Note that we will never have $x_{ij} + z_{ij} \geq 2$, as we would not perform a reinforcing step unless $x_{ij} < 1$, from which $z_{ij} < 1$ follows. Thus, to perform a cost reinforcing step we must have $y_i \leq 2$. It follows that the total number of cost reinforcing steps performed for j with optimum assignment i is at most $d\Lambda c_i \log 2m$. If we sum this over all active machines i in the optimum solution (and observe that the optimum solution must not exceed the budget B) we get $\sum_j \hat{r}_j \leq dB\Lambda \log 2m$. Combining the equations along with the assumption $n \geq 2$ gives the lemma. □

Lemma 3. *If we set* $a = 1+\frac{1}{2\log nm}$*, then the final value of the potential function* $\Phi = \sum_i \sum_k a^{\ell_i^k}$ *after all jobs have arrived is at most* $3md + 2dB$*.*

Proof. Initially, the potential function equals md (since all loads $\ell_i^k = 0$). The function Φ increases with each reinforcement step, so by Lemmas 1 and 2, the final value is $\Phi \le md + \frac{a-1}{\Lambda} \sum_j r_j \le md + \frac{a-1}{\Lambda}(\Lambda \log(nm)(\sum_i \sum_k a^{\ell_i^k} + 2dB) + n(1 + \log mn))$. For $a = 1 + \frac{1}{2\log nm}$, this implies that the final value is $\Phi \le 3md + 2dB$ (we can normalize Λ to anything, so we choose $\Lambda = \Theta(n)$). □

Theorem 2. *For* $a = 1+\frac{1}{2\log nm}$*, we have* $y_i \le [\frac{\log n}{m} + 4 + 2\log(nm)\log(3md + 2dB)]$ *for all* i*.*

Proof. Applying Lemma 3 along with the value of a specified in the theorem, we know for any k that $\sum_i a^{\ell_i^k} \le 3md + 2dB$ (where ℓ_i^k is the final load on machine i). Since each term in the summation is non-negative, we can bound the ℓ_i^k values by taking the log of both sides as $\ell_i^k \le 2\log(nm)\log(3md + 2dB)$.

We observe that y_i increases at several points in the algorithm. The total increase at step two of the algorithm can be at most $\frac{\log n}{m}$. The increases at line nine can only occur if $y_i \le 2$, and will not cause y_i to exceed four (since we can ensure $1 + \frac{1}{d\Lambda c_i} \le 2$ by scaling Λ and B appropriately). The increases at line seven always increase ℓ_i^k by the same amount as y_i, so the total increase in y_i due to these steps is at most ℓ_i^k, which is bounded as above. Combining these gives the result. □

Theorem 3. *The algorithm satisfies* $\sum_i y_i c_i \le B \log n + (6md + 8dB + 4)\log nm$*.*

Proof. Initially the left side of the equation is zero. When j arrives, every y_i increases by $\frac{1}{jm}$. Since each y_i has $c_i \le B$ (otherwise we drop that machine), the total increase in cost due to this is at most $\frac{B}{j}$. Thus in total all arrivals increase the cost by at most $\sum_j \frac{B}{j} = B \log n$.

We also increase the y_i values when we perform a reinforcing step. Some y_i values increase by an additive $z_{ij} \max_k q_{ij}^k \le \frac{x_{ij}}{\Lambda c_i}$, while others increase by a multiplicative $1 + \frac{1}{d\Lambda c_i}$. The increase in cost due to additive increases is at most $\frac{\sum_i x_{ij}}{\Lambda}$, while for multiplicative increases it is at most $\sum_i \frac{y_i}{d\Lambda}$. For the multiplicative increase to happen we must have $y_i < x_{ij} + z_{ij} < 2x_{ij}$, so the multiplicative increase is at most $\frac{2\sum_i x_{ij}}{d\Lambda}$. Since each i appears in only one of the two summations, the total increase in cost is at most $\frac{2}{\Lambda}$ for each reinforcement step. The number of reinforcement steps is bounded in Lemma 2 along with Lemma 3 (note that we normalize Λ as in Lemma 3). Combining these gives the result. □

We can normalize B to whatever value we like. Given the expressions for the competitive ratio on the load and the cost, it is natural to set $B = \Theta(m)$. This will guarantee a competitive ratio of $O(d \log nm)$ on the cost, and a competitive ratio of $O(\log(md)\log(nm))$ on the load.

2.3 Rounding

We now show how to round our fractional solution to an integral solution. Our integral solution is $O(\log(nm)\log(md))$-competitive on the load with high probability and $O(d\log^2 nm)$-competitive on cost. Suppose we have some online fractional algorithm which guarantees that the values of x_{ij} and y_i never decrease, and maintains inequalities from the linear program in Section 2.1 except that it relaxes the first inequality to $0 \leq y_i \leq \rho_A$ and the last inequality to $\sum_{i=1}^{m} y_i c_i \leq B\rho_B$. In fact, our fractional algorithm also guarantees that $x_{ij} \leq 1$. We will show that this can be rounded in an online manner to produce integral $\hat{x}_{ij}$ and $\hat{y}_i$. Observe that our algorithm from Section 2.1 will satisfy the constraints with $\rho_A = O(\log(nm)\log(md))$ and $\rho_B = O(d\log nm)$.

Our rounding procedure is as follows. For each machine i, we compute a uniformly random $r_i \in [0,1]$. We set $\hat{y}_i \leftarrow 1$ as soon as $y_i \log 2nm \geq r_i$. We define $M(j)$ as the set of machines with $\hat{y}_i = 1$ immediately after job j arrives. For each job j, let $y_i(j) = \min\{y_i, 1\}$ immediately after job j arrives. We observe that $x_{ij} \leq y_i(j)$, since the fourth linear program equation must hold at all times and $x_{ij} \leq 1$. We define $s_j = \sum_{i \in M(j)} \frac{x_{ij}}{y_i(j)}$. If $s_j < \frac{1}{2}$, then we immediately set $\hat{y}_i \leftarrow 1$ for all machines i and recompute s_j. We select exactly one machine i from $M(j)$ to assign j, setting $\hat{x}_{ij} \leftarrow 1$ for this machine only. Each machine is selected with probability $\frac{x_{ij}}{y_i(j)s_j}$.

Lemma 4. *The probability that we ever have $s_j < \frac{1}{2}$ after any j arrives is at most $\frac{1}{m}$; thus the increase in the expected total cost of the solution due to this case is at most B.*

Proof. Let $A(j)$ be the set of machines for which $y_i(j) \geq \frac{1}{\log 2nm}$. Clearly $A(j) \subseteq M(j)$ since all of these machines are active with probability one. Observe that if $\sum_{i \in A(j)} x_{ij} \geq \frac{1}{2}$, then $s_j \geq \frac{1}{2}$ with probability 1. Hence, we consider the case when $\sum_{i \in A(j)} x_{ij} < \frac{1}{2}$. Since $\sum_i x_{ij} \geq 1$, it follows that $\sum_{i \notin A(j)} x_{ij} > \frac{1}{2}$. The actual value of s_j will depend on the random choices of r_i, since s_j is computed by summing over only the active machines. We can write the equation $s_j \geq \sum_{i \notin A(j)} \frac{x_{ij}}{y_i(j)} \hat{y}_i$. Since $i \notin A(j)$, we can guarantee $E[\hat{y}_i] = y_i(j)\log 2nm$, implying $E[s_j] \geq \frac{1}{2}\log 2nm$.

The value of s_j is a sum of independent Bernoulli variables, each of which has value at most 1 (since $x_{ij} \leq y_i(j)$). Even though the variables range in between 0 and 1, we can still apply Chernoff type bounds to conclude: $P\left[s_j < \frac{1}{2}\right] \leq P\left[s_j < (1-\frac{1}{2})E[s_j]\right] \leq \left(\frac{e^{-.5}}{.5^{.5}}\right)^{E[s_j]} \leq \sqrt{\frac{2}{e}}^{\frac{1}{2}\log 2nm} \leq \frac{1}{nm}$ (assuming the base of the logarithm is a sufficiently small constant). Applying the union bound, we can sum this over all j and conclude that the probability of ever having $s_j < \frac{1}{2}$ for any j is less than $\frac{1}{m}$. Hence, the increase in expected total cost due to this case is at most $\frac{1}{m}\sum_i c_i \leq B$. □

Lemma 5. *Every job is assigned to exactly one active machine. The expected total cost of the solution is bounded by $E[\sum_i \hat{y}_i c_i] \leq B\rho_B \log 2nm$.*

Proof. The rounding scheme assigns each job to exactly one active machine, and the set of active machines only grows over time as the y_i values are non-decreasing. The manner in which the $\hat{y}_i$ are determined along with the inequality $\sum_i y_i c_i \leq B\rho_B$ produce the cost bound (since $E[\hat{y}_i] \leq y_i \log 2nm$). Note that the additional expected cost as specified in Lemma 4 in the case that $s_j < \frac{1}{2}$ is negligible and does not change the competitive ratio asymptotically. □

Lemma 6. *For any active machine i with $\hat{y}_i = 1$ and dimension k, with high probability the total load of $\sum_j \hat{x}_{ij} p_{ij}^k$ is at most $\Lambda[2\rho_\Lambda + 4\log m + \beta \log(md)]$ where β is a suitably chosen constant.*

Proof. Consider active machine i. Each job j is assigned to this machine with probability $\frac{x_{ij}}{y_i(j)s_j}$. The total load of the machine on dimension k is $\sum_j \hat{x}_{ij} p_{ij}^k$; we will assume that $p_{ij}^k \leq \Lambda$ since otherwise $x_{ij} = 0$ and thus $\hat{x}_{ij} = 0$. The problem here is that the $y_i(j)$ values change over time; if we could replace them all with the final y_i then we could use the linear program inequalities to conclude that the expected load is at most $\rho_\Lambda \Lambda$ and then apply Chernoff bounds.

Instead, we define phase α to consist of those times when $\frac{2^\alpha}{m} \leq y_i(j) < \frac{2^{\alpha+1}}{m}$, for each $0 \leq \alpha \leq \log m$. Even though $y_i > 1$ is possible, we will never have $y_i(j) > 1$ so every j arrives in some phase. Let $J(\alpha)$ be the jobs which arrive during phase α. For $\alpha < \log m$ we have: $E[\sum_{j \in J(\alpha)} \hat{x}_{ij} p_{ij}^k] \leq \sum_{j \in J(\alpha)} \frac{x_{ij}}{y_i(j)s_j} p_{ij}^k \leq \sum_{j \in J(\alpha)} 2x_{ij} p_{ij}^k \frac{m}{2^\alpha}$. However, we know $\sum_{j \in J(\alpha)} x_{ij} p_{ij}^k \leq \Lambda \frac{2^{\alpha+1}}{m}$ for any phase except the last, so we can apply this inequality to conclude $E[\sum_{j \in J(\alpha)} \hat{x}_{ij} p_{ij}^k] \leq 4\Lambda$. Thus the expected total load is at most $4\Lambda \log m$ from all phases but the last. For the last phase, the expected load is at most $2\rho_\Lambda \Lambda$, since $y_i(j) = 1$ throughout. We observe that the loads are sums of Bernoulli variables with values at most Λ, so we can apply Chernoff bounds to show that with high probability the load will not exceed its mean plus $\beta\Lambda \log(md)$ on any dimension of any machine. □

2.4 Offline Scheme and Lower Bound

For the offline case, we can simply solve the linear program, so $\rho_\Lambda = \rho_B = 1$. Instead of rounding up when $y_i \log 2nm \geq r_i$, we can round up when $y_i \log 2n \geq r_i$. This increases the probability of the bad event where some $s_j < \frac{1}{2}$ to be a small constant instead of $\frac{1}{m}$, but we can simply discard our solution and retry whenever this occurs. This means we can get a solution of cost $O(\log n)$ times B, such that a fractional solution exists which exceeds Λ by at most a constant factor on any machine. We can then convert our fractional solution to an integer solution using the rounding approach of the Generalized Assignment Problem [26]. This attains roughly the same bounds as [17] for the offline case with a substantially simpler rounding scheme.

There is a simple example which indicates that no deterministic approach to this problem can succeed. We simply give a series of requests each of which can run on every machine *except the ones where the preceding requests were scheduled.* This forces a deterministic algorithm to pay for all m machines, whereas the

offline optimum could activate only a single machine. The Online Set Cover result of Alon et al. [1] got around this by presuming that we know in advance the set of elements (jobs) which *might* be requested in the future, and allowed the competitive ratio to depend on the size of this set. This approach seems less reasonable for our problem than theirs, but in any case a modification of their derandomization should work in the same model.

3 Machine Activation Variants

We study four versions of online load balancing with activation costs. For the version where both Λ and B are given, Section 2 gives an algorithm that is $O(\log(nm)\log(md))$-competitive on the load and $O(d\log^2(nm))$-competitive on cost. We also consider variants where either B is not given up front or Λ is not given up front (or both are not given). Due to space constraints, the following proofs can be found in the full version of our paper. Our positive results are obtained by guessing the value of the objective function to be optimized in an online manner using doubling techniques combined with our algorithm from Section 2.3. The logarithmic dependence on the optimum load (or budget) in some of the results is undesirable, and we observe that it would not occur in the offline setting (where we can discard the solution of previous phases and start over with each new phase). However, we can show that this dependence is necessary for a fully online algorithm.

Theorem 4. *For the version where we are given a budget B and asked to minimize load Λ, we can produce a solution which spends at most $B\rho_B$ and has competitive ratio ρ_Λ on the load against the optimum offline with budget B. Here $\rho_\Lambda = O(\log(md)\log(nm))$ but $\rho_B = O(d\log^2(nm)\log\Lambda^*)$ where Λ^* is the optimum load (assuming we have a lower bound on the load of one; otherwise it is the ratio of maximum to minimum possible non-zero load).*

Theorem 5. *For the version where we are given load Λ and asked to minimize the budget B, we can produce a solution with load at most $\rho_\Lambda\Lambda$ which has competitive ratio ρ_B against the optimum offline which does not exceed load Λ. Here $\rho_B = O(d\log^2 nm)$ but $\rho_\Lambda = O(\log(md)\log(nm)\log B^*)$ where B^* is the ratio of the optimum budget to the minimum non-zero cost of a machine.*

Theorem 6. *Consider the version where we are given a budget B and asked to minimize load Λ. Suppose that the actual optimum load is Λ^*, and we are to guarantee that we spend a budget at most $\rho_B B$ and obtain load at most ρ_Λ times optimum. Then $\rho_B = \Omega(\min\{\frac{\log\Lambda^*}{\log\log\Lambda^*+\log\rho_\Lambda}, m\})$.*

Theorem 7. *If a deterministic algorithm guarantees to be ρ_Λ-competitive on the makespan while using at most $\rho_B B$ budget, then the algorithm necessarily has $\rho_B = \Omega\left(\frac{\log\Lambda^*(\log n-\log\log m-\log\log\Lambda^*)(\log m-\log\log\Lambda^*)}{(\log\rho_\Lambda+\log\log n+\log\log m+\log\log\Lambda^*)(\log\log n+\log\log m)}\right)$.*

Theorem 8. *Consider the version where we are given a load Λ and asked to minimize the cost B. Suppose that the optimum cost is B^* to stay within load Λ. We are to guarantee load at most $\Lambda\rho_\Lambda$ and cost at most ρ_B times B^*. Then* $\rho_\Lambda = \Omega(\min\{\frac{\log B^*}{\log 2\rho_B}, m\})$.

Theorem 9. *There is an $O(d \log^2 nm)$-competitive algorithm for minimizing a linear combination of the cost B and load Λ, namely $c_B B + c_\Lambda \Lambda$. Here, the coefficients c_B, c_Λ are constants and $c_B B + c_\Lambda \Lambda$ is the value of the objective function on a schedule with energy cost B and makespan Λ.*

4 Vector Load Balancing

We give an $O(\log d)$-competitive algorithm to minimize the makespan, improving over the offline $O(\log^2 d)$-approximation in [9] (for arbitrary d). Note that [9] shows that even in the offline case, no constant approximation is possible unless NP = ZPP. We extend our algorithm to be simultaneously $O(\log d)$-competitive on energy, provided we know $\Lambda_{max} = \max_{k,j} p_j^k$ in advance (p_j^k is the load of job j on component k). We show that this additional information is necessary.

4.1 Minimizing the Makespan

The algorithm of [2] depends heavily on maintaining an estimate Λ of the optimum value. We use the maximum coordinate of any single job $\Lambda_{max} = \max_{k,j} p_j^k$ and the load induced by placing all jobs on one machine $\Lambda_{tot} = \max_k \sum_j p_j^k$ as lower bounds in Algorithm 2. We run in phases where for each phase, we use an estimate Λ of the optimum makespan. If Λ is too small, we adjust and start a new phase. We also make use of $a = 1 + \frac{1}{\gamma}$ for some $\gamma > 1$. Let $\ell_i^{kx}(j)$ be the load normalized by Λ on component k of machine i during phase x after all jobs up through j are assigned (we sometimes omit j if the context is clear). Let p_j^k be the load induced on dimension k by job j, and $q_j^k = p_j^k/\Lambda$. Lemma 7 is proved in the full version of our paper (this is a modification of the proof in [2]).

Lemma 7. *In Algorithm 2, during each phase x:* $\sum_{k=1}^{d} \sum_{i=1}^{m} a^{\ell_i^{kx}} (\gamma - 1) \leq \gamma md$.

At this point, we can follow [2] and use Lemma 7 for an $O(\log dm)$ bound. We will improve our competitive ratio by exploiting the identical nature of our machines. Let $\mu^x = \max_{i,k} \ell_i^{kx}$.

Lemma 8. *For all machines i, i' and phases x, we have $\sum_{k=1}^{d} a^{\ell_i^{kx}(f)} \leq d + \frac{\mu^x}{\gamma} \sum_{k=1}^{d} a^{\ell_{i'}^{kx}(f)}$, where f is the final job of phase x.*

Proof. Let J_i be the set of jobs given to machine i during phase x. If $j \in J_i$, machine i must minimize $\Delta_i(j) = \sum_{k=1}^{d} a^{\ell_i^{kx}(j-1)+q_j^k} - a^{\ell_i^{kx}(j-1)}$. Hence, for any

```
1 Initialize Λ ← 1, x ← 0, ℓ_i^{kt} ← 0 for all i, k, t
2 while jobs j arrive do
3     Update Λ_max and Λ_tot
4     if Λ < max{Λ_max, (1/m)Λ_tot} then
5         End phase x; let x ← ⌈log_2 max{Λ_max, (1/m)Λ_tot}⌉, Λ ← 2^x
6     Place job j on machine s = argmin_i Σ_{k=1}^{d} [a^{ℓ_i^{kx}+q_j^k} − a^{ℓ_i^{kx}}]; let
      ℓ_s^{kx} ← ℓ_s^{kx} + q_j^k for all k
```

Algorithm 2. Assign-Jobs

$i', \sum_{k=1}^{d}(a^{\ell_i^{kx}(f)}-1) = \sum_{j\in J_i}\Delta_i(j) \le \sum_{j\in J_i}\Delta_{i'}(j) \le \sum_{k=1}^{d} a^{\ell_{i'}^{kx}(f)} \sum_{j\in J_i}(a^{q_j^k}-1)$. Since $p_j^k \le \Lambda$, we have $0 \le q_j^k \le 1$ and thus $\gamma(a^{q_j^k}-1) \le q_j^k$. This allows us to bound $\sum_{k=1}^{d} a^{\ell_{i'}^{kx}(f)} \sum_{j\in J_i}(a^{q_j^k}-1)$ by

$$\sum_{k=1}^{d} a^{\ell_{i'}^{kx}(f)} \sum_{j\in J_i} \frac{q_j^k}{\gamma} = \sum_{k=1}^{d} a^{\ell_{i'}^{kx}(f)} \frac{\ell_i^{kx}(f)}{\gamma} \le \frac{\mu^x}{\gamma} \sum_{k=1}^{d} a^{\ell_{i'}^{kx}(f)}.$$

Putting our inequalities together and rearranging terms finishes the proof. □

Theorem 10. *Algorithm 2 is $O(\log d)$-competitive on the makespan.*

Proof. Fix phase x and let s and s' be the machines with maximal and minimal $\sum_{k=1}^{d} a^{\ell_i^{kx}}$ (respectively) at the end of x. Combining Lemmas 7 and 8 gives

$$a^{\mu^x} \le \sum_{k=1}^{d} a^{\ell_s^{kx}(f)} \le d + \frac{\mu^x}{\gamma} \sum_{k=1}^{d} a^{\ell_{s'}^{kx}(f)} \le d + \frac{\mu^x}{\gamma}\left(\frac{\gamma d}{\gamma-1}\right) = d + \frac{\mu^x d}{\gamma-1}.$$

Taking the logarithm of both sides gives $\mu^x \le \log_a\left(\frac{d}{\gamma-1}\right) + \log_a(\mu^x+\gamma-1)$. Thus, $\mu^x - \log_a(\mu^x+\gamma-1) = O(\log d)$. Note that if for some constant c we have $c - \log_a(c+\gamma-1) = \frac{c}{2}$, then for all $\mu^x \ge c$, we have $\mu^x = O(\log d)$. The makespan during phase x is $2^x O(\log d)$. For our last phase g, our total load is at most $2^{g+1}O(\log d)$. Since $g = \lceil \log(\max\{\Lambda_{max}, \frac{1}{m}\Lambda_{tot}\})\rceil$ and $\max\{\Lambda_{max}, \frac{1}{m}\Lambda_{tot}\}$ is a lower bound on optimum, our algorithm is $O(\log d)$-competitive. □

4.2 Simultaneously Minimizing Energy

Given the value of Λ_{max} in advance, we compress all jobs onto a small number of machines, then gradually open up more machines as our estimate of Λ_{tot} increases. Algorithm 3 does this with virtual machines. As there are at most $2m$ virtual machines in total, we identify two virtual machines with each real machine. We prove Theorem 12 in the full version of our paper, which establishes that advance knowledge of Λ_{max} (or some comparable advance knowledge) is necessary to have a competitive ratio independent of m.

```
Initialize M ← 1
while M < m do
    Run algorithm Assign-Jobs on M new virtual machines until Λ_tot > MΛ_max
    M ← min{2M, m}
while jobs are still arriving do
    Run algorithm Assign-Jobs on all real machines ignoring previous loads
```

Algorithm 3. Power-and-Makespan

Theorem 11. *Algorithm 3 is $O(\log d)$-competitive on both makespan and power.*

Proof. Within a call to Assign-Jobs, we guarantee that $\Lambda_{tot} \leq M\Lambda_{max}$. Thus by Theorem 10, we place a load of at most $\Lambda_{max}O(\log d)$ on any virtual machine. Then after all jobs are placed, the load of any real machine is at most the sum of the loads of two virtual machines (each at most $\Lambda_{max}O(\log d)$), plus any load placed by the last instance of Assign-Jobs when $M = m$. Thus the makespan is at most $2\Lambda_{max}O(\log d) + \frac{1}{m}\Lambda_{tot}O(\log d)$. We observe that at most $2M$ machines have non-zero load, so the total power is at most $4M\Lambda_{max}O(\log d) + 2\Lambda_{tot}O(\log d)$. Optimum power is Λ_{tot} and the algorithm guarantees $\Lambda_{tot} > \frac{M}{2}\Lambda_{max}$, which completes the proof. □

Theorem 12. *Suppose we have an online algorithm which is α-competitive on the makespan and β-competitive on energy. Then $\beta \geq \frac{1}{2\alpha}\min(d, \log_{2\alpha} m)$.*

References

1. Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., Naor, J.: The online set cover problem. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing (2003)
2. Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., Waarts, O.: On-line load balancing with applications to machine scheduling and virtual circuit routing. In: Proceedings of the 25th Annual ACM Symposium on Theory of Computing (1993)
3. Awerbuch, B., Azar, Y., Grove, E., Kao, M., Krishnan, P., Vitter, J.: Load balancing in the l_p norm. In: Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (1995)
4. Azar, Y., Bhaskar, U., Fleischer, L., Panigrahi, D.: Online mixed packing and covering. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (2013)
5. Azar, Y., Cohen, I.R., Kamara, S., Shepherd, B.: Tight bounds for online vector bin packing. In: Proceedings of the 45th Annual ACM Symposium on Theory of Computing (2013)
6. Bansal, N., Caprara, A., Sviridenko, M.: Improved approximation algorithms for multidimensional bin packing problems. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (2006)
7. Bartal, Y., Fiat, A., Karloff, H., Vohra, R.: New algorithms for an ancient scheduling problem. In: Proceedings of the 24th Annual ACM Symposium on Theory of Computing (1992)

8. Buchbinder, N., Naor, J.: Fair online load balancing. In: Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures (2006)
9. Chekuri, C., Khanna, S.: On multi-dimensional packing problems. In: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (1999)
10. DeWitt, D., Gray, J.: Parallel database systems: The future of high performance database systems. Communications of the ACM (1992)
11. Garofalakis, M., Ioannidis, Y.: Scheduling issues in multimedia query optimization. ACM Computing Surveys (1995)
12. Garofalakis, M., Ioannidis, Y.: Multi-dimensional resource scheduling for parallel queries. In: Proceedings of 1996 ACM SIGMOD International Conference on Management of Data (1996)
13. Garofalakis, M., Ioannidis, Y.: Parallel query scheduling and optimization with time- and space-shared resources. In: Proceedings of the 23rd International Conference on Very Large Data Bases (1997)
14. Hochbaum, D., Shmoys, D.: Using dual approximation algorithms for scheduling problems: theoretical and practical results. Journal of the ACM 34(1), 144–162 (1987)
15. Irani, S., Pruhs, K.: Algorithmic problems in power management. SIGACT News 36(2), 63–76 (2005)
16. Karp, R., Luby, M., Marchetti-Spaccamela, A.: A probabilistic analysis of multidimensional bin packing problems. In: Proceedings of the 16th Annual ACM Symposium on Theory of Computing (1984)
17. Khuller, S., Li, J., Saha, B.: Energy efficient scheduling via partial shutdown. In: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (2010)
18. Kou, L., Markowsky, G.: Multidimensional bin packing algorithms. IBM Journal of Research and Development (1977)
19. Lavi, R., Swamy, C.: Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. In: Proceedings of the 8th Annual ACM Conference on Electronic Commerce (2007)
20. Li, J., Khuller, S.: Generalized machine activation problems. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (2011)
21. Lo, S.-T., Chen, R.-M., Huang, Y.-M.: Multi-constraint system scheduling using dynamic and delay ant colony system. In: Okuno, H.G., Ali, M. (eds.) IEA/AIE 2007. LNCS (LNAI), vol. 4570, pp. 655–664. Springer, Heidelberg (2007)
22. Moore, J., Chase, J., Ranganathan, P., Sharma, R.: Making scheduling "cool": temperature-aware workload placement in data centers. In: Proceedings of the USENIX Annual Technical Conference (2005)
23. Pruhs, K., Sgall, J., Torng, E.: Online scheduling. In: Handbook of Scheduling: Algorithms, Models, and Performance Analysis (2004)
24. Rofouei, M., Stathopoulos, T., Ryffel, S., Kaiser, W., Sarrafzadeh, M.: Energy aware high performance computing with graphical processing units. In: Proceedings of USENIX 1st Workshop on Power Aware Computing Systems (2008)
25. Ryffel, S., Stathopoulos, T., McIntire, D., Kaiser, W., Thiele, L.: Accurate energy attribution and accounting for multi-core systems. Technical Report 67, Center for Embedded Network Sensing (2009)
26. Shmoys, D., Tardos, E.: An approximation algorithm for the generalized assignment problem. Math. Program. 62(3), 461–474 (1993)
27. Stathopoulos, T., McIntire, D., Kaiser, W.: The energy endoscope: Real-time detailed energy accounting for wireless sensor nodes. In: Proceedings of the 7th Annual International Conference on Information Processing in Sensor Networks (2008)

A New Regularity Lemma and Faster Approximation Algorithms for Low Threshold Rank Graphs

Shayan Oveis Gharan[1,*] and Luca Trevisan[2,**]

[1] Department of Management Science and Engineering, Stanford University
shayan@stanford.edu
[2] Department of Computer Science, Stanford University
trevisan@stanford.edu

Abstract. Kolla and Tulsiani [KT07, Kol11] and Arora, Barak and Steurer [ABS10] introduced the technique of *subspace enumeration*, which gives approximation algorithms for graph problems such as unique games and small set expansion; the running time of such algorithms is exponential in the *threshold-rank* of the graph.

Guruswami and Sinop [GS11, GS12], and Barak, Raghavendra, and Steurer [BRS11] developed an alternative approach to the design of approximation algorithms for graphs of bounded threshold-rank, based on semidefinite programming relaxations in the Lassere hierarchy and on novel rounding techniques. These algorithms are faster than the ones based on subspace enumeration and work on a broad class of problems.

In this paper we develop a third approach to the design of such algorithms. We show, constructively, that graphs of bounded threshold-rank satisfy a *weak Szemeredi regularity lemma* analogous to the one proved by Frieze and Kannan [FK99] for dense graphs. The existence of efficient approximation algorithms is then a consequence of the regularity lemma, as shown by Frieze and Kannan.

Applying our method to the Max Cut problem, we devise an algorithm that is faster than all previous algorithms, and is easier to describe and analyze.

1 Introduction

Kolla and Tulsiani [KT07, Kol11] and Arora, Barak and Steurer [ABS10] proved that the Unique Games problem can be approximated efficiently if the adjacency matrix of a graph associated with the problem has few large eigenvalues; they show that, for every optimal solution, its indicator vector is close to the subspace spanned by the eigenvectors of the large eigenvalues, and one can find a solution close to an optimal one by enumerating an ϵ-net for such a subspace.

* Supported by a Stanford Graduate Fellowship.
** This material is based upon work supported by the National Science Foundation under grant No. CCF 1017403.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 303–316, 2013.

Such *subspace enumeration* algorithm runs in time exponential in the dimension of the subspace, which is the number of large eigenvalues; such a parameter is called the *threshold rank* of the graph. Arora, Barak and Steurer show that the subspace enumeration algorithm can approximate other graph problems, in regular graphs, in time exponential in the threshold rank, including the Uniform Sparsest Cut problem, the Small-Set Expansion problem and the Max Cut problem. We remark that the subspace enumeration algorithm does not improve the 0.878 approximation guarantee of Goemans, Williamson [GW95], but it finds a solution of approximation factor $1 - O(\epsilon)$ if the optimum cuts at least $1 - \epsilon$. fraction of edges.

Barak, Raghavendra and Steurer [BRS11] and Guruswami and Sinop [GS11, GS12, GS13] developed an alternative approach to the design of approximation algorithms running in time exponential in the threshold rank. Their algorithms are based on solving semidefinite programming relaxations from the Lasserre hierarchy and then applying sophisticated rounding schemes. The advantage of this approach is that it is applicable to a more general class of graph problems and constraint satisfaction problems, that the approximation guarantee has a tighter dependency on the threshold used in the definition of threshold rank and that, in same cases, the algorithms have a running time of $f(k, \epsilon) \cdot n^{O(1)}$ where k is the threshold rank and $1 \pm \epsilon$ is the approximation guarantee, instead of the running time of $n^{O(k)}$ which follows from an application of the subspace enumeration algorithm for constant ϵ.

In this paper we introduce a third approach to designing algorithms for graphs of bounded threshold rank, which is based on proving a *weak Szemeredi regularity lemma* for such graphs.

The regularity lemma of Szemeredi [Sze78] states that every dense graph can be well approximated by the union of a constant number of bipartite complete subgraphs; the constant, however, has a tower-of-exponentials dependency on the quality of approximation. Frieze and Kannan [FK96, FK99] prove what they call a *weak regularity lemma*, showing that every dense graph can be approximated up to an error ϵn^2 in the cut norm by a linear combination of $O(1/\epsilon^2)$ cut matrices (a cut matrix is a bipartite complete subgraph) with bounded coefficients. Frieze and Kannan also show that such an approximation can be constructed "implicitly" in time polynomial in $1/\epsilon$ and that, for a weighted graph which is a linear combination of σ cut matrices, several graph problems can be approximated in time $\exp(\tilde{O}(\sigma)) + \text{poly}(n)$ time. Combining the two facts one has a $\exp(\text{poly}(1/\epsilon))+\text{poly}(n)$ time approximation algorithm for many graph problems on dense graphs.

We prove that a weak regularity lemma holds for all graphs of bounded threshold rank. Our result is a proper generalization of the weak regularity lemma of Frieze and Kannan, because dense graphs are known to have bounded threshold rank[1]. For a (weighted) $G = (V, E)$ with adjacency matrix A, and diagonal matrix of vertex degrees D, $D^{-1/2}AD^{-1/2}$ is called the normalized adjacency matrix of G. If the square sum of the eigenvalues of the normalized adjacency

[1] The normalization used for dense graphs is different.

matrix outside the range $[-\epsilon/2, \epsilon/2]$ is equal to k (in particular, if there are at most k such eigenvalues), then we show that there is a linear combination of $O(k/\epsilon^2)$ cut matrices that approximate A up to $2\epsilon|E|$ in cut norm; furthermore, such a decomposition can be found in $\text{poly}(n, k, 1/\epsilon)$ time. (See Theorem 1 below.) Our regularity lemma, combined with an improvement of the Frieze-Kannan approximation algorithm for graphs that are linear combination of cut matrices, gives us algorithms of running time $2^{\tilde{O}(k^{1.5}/\epsilon^3)} + \text{poly}(n)$ for several graph problems on graphs of threshold rank k, providing an additive approximation of $2\epsilon|E|$. In problems such as Max Cut in which the optimum is $\Omega(|E|)$, this additive approximation is equivalent to a multiplicative approximation.

We remark that there are several generalizations of the weak regularity lemma to the matrices that are not necessarily dense, e.g., [DKKV05, CCF09], but to the best of our knowledge, none of these generalizations include matrices of low threshold rank. For example, Coja-Oghlan, Cooper and Frieze [CCF09] consider sparse matrices that have a suitable boundedness property. For $S, T \subseteq V$, let the density of sub-matrix $A(S,T)$ be the ratio of sum of the entries in $A(S,T)$ to $|S| \cdot |T|$. Coja-Oghlan et al. [CCF09] generalizes weak regularity lemma to matrices where the density of each $A(S,T)$ is within a constant factor, C, of the density of A, for any S, T such that $|S|, |T| \geq \Omega(n/2^{C^2})$. But, if A represents the adjacency matrix of a graph that is a union of a constant number of constant degree expanders, then although A has a bounded threshold rank, it doesn't satisfy the boundedness property.

Table 1. A comparison between previous algorithms applied to Max Cut and our algorithm. [BRS11] needs to solve r rounds of Lasserre hierarchy, for $r = O(k/\epsilon^4)$.

Reference	Running time	Parameter k
[BRS11]*	$2^{O(k/\epsilon^4)} \cdot \text{poly}(n)$	# of eigenvalues not in range $[-c \cdot \epsilon^2, c \cdot \epsilon^2]$, $c > 0$
[GS11]	$n^{O(k/\epsilon^2)}$	# of eigenvalues $\leq -\epsilon/2$
[GS12]	$2^{k/\epsilon^3} \cdot n^{O(1/\epsilon)}$	# of eigenvalues $\leq -\epsilon/2$
this paper	$2^{\tilde{O}(k^{1.5}/\epsilon^3)} + \text{poly}(n)$	sum of squares of eigenvalues not in range $[-\epsilon/8, \epsilon/8]$

Table 1 gives a comparison between previous algorithms applied to Max Cut and our algorithm. Unlike the previous algorithms, our algorithm rounds the solution to a fixed size LP, as opposed to a SDP hierarchy. The advantages over previous algorithms, besides the simplicity of the algorithm, is a faster running time and the dependency on a potentially smaller theshold-rank parameter, because the running time of our algorithm depends on the *sum of squares* of eigenvalues outside of a certain range, rather than the number of such eigenvalues. (recall that the eigenvalues of $D^{-1/2}AD^{-1/2}$ are in the range $[-1, 1]$.)

We now give a precise statement of our results, after introducing some notation.

2 Statement of Results

2.1 Notations

Let $G = (V, E)$ be a (weighted) undirected graph with $n := |V|$ vertices. Let A be the adjacency matrix of G. For a vertex $u \in V$, let $d(u) := \sum_{v \sim u} A(u, v)$ be the degree of u. For a set $S \subset V$, let the *volume* of S be the summation of vertex degrees in S, $d(S) = \sum_{v \in S} d(v)$, and and let $m := d(V)$. Let D be the diagonal matrix of degrees. For any matrix $M \in \mathbb{R}^{V \times V}$, we use $M_{\mathcal{D}}$ to denote the symmetric matrix $D^{-1/2} M D^{-1/2}$. Observe that if G is a d-regular graph, then $M_{\mathcal{D}} = M/d$. We call $A_{\mathcal{D}}$ the *normalized adjacency matrix* of G. It is straightforward to see that all eigenvalues of $A_{\mathcal{D}}$ is contained in the interval $[-1, 1]$.

For two functions $f, g \in \mathbb{R}^V$, let $\langle f, g \rangle := \sum_{v \in V} f(v) g(v)$. Also, let $f \otimes g$ be the tensor product of f, g; i.e., the matrix in $\mathbb{R}^{V \times V}$ such that (u, v) entry is $f(u) \cdot g(v)$. For a function $f \in \mathbb{R}^V$, and $S \subseteq V$ let $f(S) := \sum_{v \in S} f(v)$.

For a set $S \subseteq V$, let $\mathbf{1}_S$ be the indicator function of S, and let

$$d_S(v) := \begin{cases} d(v) & v \in S \\ 0 & \text{otherwise.} \end{cases}$$

For any two sets $S, T \subseteq V$, and $\alpha \in \mathbb{R}$, we use the notation $\mathrm{CUT}(S, T, \alpha) := \alpha(d_S \otimes d_T)$ to denote the matrix corresponding to the cut (S, T), where (u, v) entry of the matrix is $\alpha d(u) \cdot d(v)$ if $u \in S, v \in T$ and zero otherwise. We remark that $\mathrm{CUT}(S, T, \alpha)$ is not necessarily a symmetric matrix.

Definition 1 (Matrix Norms). *For a matrix $M \in \mathbb{R}^{V \times V}$, and $S, T \subseteq V$, let*

$$M(S, T) := \sum_{u \in S, v \in T} M_{u,v}.$$

The Frobenius norm and the cut norm are defined as follows:

$$\|M\|_F := \sqrt{\sum_{u,v} M_{u,v}^2},$$
$$\|M\|_C := \max_{S,T \subseteq V} |M(S, T)|$$

Definition 2 (Sum-Squares Threshold Rank). *For any unweighted graph G, with normalized adjacency matrix $A_{\mathcal{D}}$, let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $A_{\mathcal{D}}$ with the corresponding eigenfunctions $f_1, \ldots, f_n$. For $\delta > 0$, the δ sum-squares threshold rank of A is defined as*

$$t_\delta(A_{\mathcal{D}}) := \sum_{i : |\lambda_i| > \delta} \lambda_i^2.$$

Also, the δ threshold approximation of A is defined as,

$$T_\delta(A_{\mathcal{D}}) := \sum_{i : |\lambda_i| > \delta} \lambda_i f_i \otimes f_i.$$

2.2 Matrix Decomposition Theorem

The following matrix decomposition theorem is the main technical result of this paper.

Theorem 1. *For any graph G, and $\epsilon > 0$, let $k := t_{\epsilon/2}(A_{\mathcal{D}})$. There is a algorithm that writes A as a linear combination of cut matrices, $W^{(1)}, W^{(2)}, \ldots, W^{(\sigma)}$, such that $\sigma \leq 16k/\epsilon^2$, and*

$$\left\| A - W^{(1)} - \ldots - W^{(\sigma)} \right\|_C \leq \epsilon m,$$

where each $W^{(i)}$ is a cut matrix $\mathrm{CUT}(S, T, \alpha)$, for some $S, T \subseteq V$, such that $|\alpha| \leq \sqrt{k}/m$. The running time of the algorithm is polynomial in $n, k, 1/\epsilon$.

2.3 Algorithmic Applications

Our main algorithmic application of Theorem 1 is the following theorem that approximates any cut on low threshold rank graphs with a running time $2^{\tilde{O}(k^{1.5}/\epsilon^3)} + \mathrm{poly}(n)$.

Theorem 2. *Let $G = (V, E)$, and for a given $0 < \epsilon$, let $k := t_{\epsilon/8}(A_{\mathcal{D}})$. There is a randomized algorithm such that for either of maximum cut, the minimum cut problem on sets of volume Γ, in time $2^{\tilde{O}(k^{1.5}/\epsilon^3)} + \mathrm{poly}(n)$, with constant probability finds a set S such that $|d(S) - \Gamma| \leq \epsilon m$ and for any S^* of size $d(S^*) = \Gamma$,*

$$A(S, \overline{S}) \geq A(S^*, \overline{S^*}) - \epsilon m$$

if it is a maximization problem, otherwise,

$$A(S, \overline{S}) \leq A(S^*, \overline{S^*}) + \epsilon m.$$

We can use the above theorem to provide a PTAS for maximum cut, maximum bisection, and minimum bisection problems.

Corollary 1. *Let $G = (V, E)$, and for a given $\epsilon > 0$, let $k := t_{\epsilon/8}(A_{\mathcal{D}})$. There is a randomized algorithm that in time $2^{\tilde{O}(k^{1.5}/\epsilon^3)} + \mathrm{poly}(n)$ finds an ϵm additive approximation of the maximum cut.*

Proof. We can simply guess the size of the optimum within an $\epsilon m/2$ additive error and then use Theorem 2. □

Corollary 2. *Let $G = (V, E)$, and for a given $\epsilon > 0$, let $k := t_{\epsilon/8}(A_{\mathcal{D}})$. For any of the maximum bisection and minimum bisection problems, there is a randomized algorithm that in time $2^{\tilde{O}(k^{1.5}/\epsilon^3)} + \mathrm{poly}(n)$ finds a cut $(S, \overline{S})$ such that $|d(S) - m/2| \leq \epsilon m$ and that $A(S, \overline{S})$ provides an ϵm additive approximation of the optimum.*

Proof. For the maximum/minimum bisection the optimum must have size $m/2$. So we can simply use Theorem 2 with $\Gamma = m/2$. □

We remark although in the literature a bisection is typically defined as the cut with equal number of vertices in the both sides, above corollary finds a cut with (approximately) equal volume. This is a limitation of spectral algorithms (c.f. Cheeger's inequality for finding the minimum bisection). Nonetheless, the applications are very similar (e.g. we can use above corollary in divide and conquer algorithms to partition the graph into small pieces with few edges in between).

3 Regularity Lemma for Low Threshold Rank Graphs

In this section we prove Theorem 1. The first step is to approximate A by a low rank matrix B. In the next lemma we construct B such that the value of any cut in A is approximated within an small additive error in B.

Lemma 1. *Let A be the adjacency matrix of G. For $0 \leq \delta < 1$, let*

$$B := D^{1/2} T_\delta(A_{\mathcal{D}}) D^{1/2}.$$

Then, $\|A - B\|_C \leq \delta m$.

Proof. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $A_{\mathcal{D}}$, with the corresponding eigenfunctions $f_1, \ldots, f_n$. For any $S, T \subseteq V$, we have

$$\begin{aligned}
\langle \mathbf{1}_S, (A-B)\mathbf{1}_T \rangle &= \langle D^{1/2}\mathbf{1}_S, (A-B)_{\mathcal{D}} D^{1/2}\mathbf{1}_T \rangle \\
&= \langle \sqrt{d_S}, (A_{\mathcal{D}} - T_\delta(A_{\mathcal{D}}))\sqrt{d_T} \rangle \\
&\leq \delta \sum_{i:|\lambda_i|\leq\delta} \langle \sqrt{d_S}, f_i \rangle \langle \sqrt{d_T}, f_i \rangle \\
&\leq \delta \sqrt{\sum_{i:|\lambda_i|\leq\delta} \langle \sqrt{d_S}, f_i \rangle^2} \cdot \sqrt{\sum_{i:|\lambda_i|\leq\delta} \langle \sqrt{d_T}, f_i \rangle^2} \\
&\leq \delta \left\| \sqrt{d_S} \right\| \left\| \sqrt{d_T} \right\| \leq \delta \left\| \sqrt{d_V} \right\|^2 = \delta m,
\end{aligned}$$

where the second inequality follows by the Cauchy-Schwarz inequality. The lemma follows by noting the fact that $\|A - B\|_C$ is the maximum of the above expression for any $S, T \subseteq V$. □

By the above lemma if we approximate B by a linear combination of cut matrices, that also is a good approximation of A. Moreover, since $t_\delta(A_{\mathcal{D}}) = t_\delta(B_{\mathcal{D}})$, B has a small sum-square threshold rank iff A has a small sum-square threshold rank.

Lemma 2. *For any graph G with adjacency matrix A, and $\delta > 0$, let $B := D^{1/2} T_\delta(A_{\mathcal{D}}) D^{1/2}$. Then,*

$$\|B_{\mathcal{D}}\|_F^2 = t_\delta(A_{\mathcal{D}}).$$

Proof. The lemma follows from the fact that the square of the frobenius norm of any matrix is equal to the summation of square of eigenvalues. If $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $A_{\mathcal{D}}$, then

$$\|B\|_F^2 = \operatorname{trace} B^2 = \sum_{|\lambda_i|>\delta} \lambda_i^2 = t_\delta(A_{\mathcal{D}})\,.$$

□

The next proposition is the main technical part of the proof of Theorem 1. We show that we can write any (not necessarily symmetric) matrix B as a linear combination of $O(\|B\|_F^2/\epsilon^2)$ cut matrices such that the cut norm of B is preserved within an additive error of ϵm. The proof builds on the existential theorem of Frieze and Kanan [FK99, Theorem 7].

Proposition 1. *For any matrix $B \in \mathbb{R}^{V\times V}$, $k = \|B_{\mathcal{D}}\|_F^2$, and $\epsilon > 0$, there exist cut matrices $W^{(1)}, W^{(2)}, \ldots, W^{(\sigma)}$, such that $\sigma \le 1/\epsilon^2$, and for all $S, T \subseteq V$,*

$$\left|\left(B - W^{(1)} - W^{(2)} - \ldots - W^{(\sigma)}\right)(S,T)\right| \le \epsilon\sqrt{k \cdot d(S) \cdot d(T)},$$

where each $W^{(i)}$ is a cut matrix $\mathrm{CUT}(S,T,\alpha)$, for some $S, T \subseteq V$, and $\alpha \in \mathbb{R}$.

Proof. Let $R^{(0)} = B$. We use the potential function $h(R) := \|R_{\mathcal{D}}\|_F^2$. We show that as long as $\|R\|_C > \epsilon\sqrt{k}m$, we can choose cut matrices iteratively while maintaining the invariant that each time the value of the potential function decreases by at least $\epsilon^2 h(B)$. Since $h(R^{(0)}) = h(B)$, after at most $1/\epsilon^2$ we obtain a good approximation of B.

Assume that after $t < 1/\epsilon^2$ iterations, $R^{(i)} = B - W^{(1)} - \ldots - W^{(i)}$. Suppose for some $S, T \subseteq V$,

$$\left|R^{(i)}(S,T)\right| > \epsilon\sqrt{h(B) \cdot d(S) \cdot d(T)}. \tag{1}$$

Choose $W^{(i+1)} = \mathrm{CUT}(S,T,\alpha)$, for $\alpha = R^{(i)}(S,T)/d(S)d(T)$, and let $R^{(i+1)} = R^{(i)} - W^{(i+1)}$. We have,

$$\begin{aligned} h(R^{(i+1)}) - h(R^{(i)}) &= \sum_{u\in S, v\in T} \frac{(R^{(i)}_{u,v} - \alpha d(u)d(v))^2 - {R^{(i)}_{u,v}}^2}{d(u)d(v)} \\ &= -2\alpha R^{(i)}(S,T) + \alpha^2 d(S)d(T) \\ &= \frac{-R^{(i)}(S,T)^2}{d(S)d(T)} \le -\epsilon^2 \cdot h(B), \end{aligned}$$

where the second to last equation follows from the definition of α, and the last equation follows from equation (1). Therefore, after at most $\sigma \le 1/\epsilon^2$ iterations, (1) must hold for all $S, T \subseteq V$. □

Although the previous proposition only proves the existence of a decomposition into cut matrices, we can construct such a decomposition efficiently using the following nice result of Alon and Naor [AN06] that gives a consant factor approximation algorithm for the cut norm of any matrix.

Theorem 3 (Alon and Naor [AN06]). *There is a polynomial time randomized algorithm such that for any given $A \in \mathbb{R}^{V\times V}$, with high probability, finds sets $S, T \subseteq V$, such that*

$$|A(S,T)| \geq 0.56 \|A\|_C .$$

Now we are ready to prove Theorem 1.

Proof of Theorem 1. Let $\delta := \epsilon/2$, and $B := D^{1/2}T_\delta(A_{\mathcal{D}})D^{1/2}$. By Lemma 1, we have that $\|A-B\|_C \leq \epsilon m/2$. So we just need to approximate B by a set of cut matrices within an additive error of $\epsilon m/2$. For a matrix R, let $h(R) := \|R_{\mathcal{D}}\|_F^2$. By Lemma 2 we have $h(B) = k$.

Let $\epsilon' := \epsilon/\sqrt{4k}$. We use the proof strategy of Proposition 1. Let $R^{(i)} = B - W^{(1)} - \ldots - W^{(i)}$. If $\left\|R^{(i)}\right\|_C \geq \epsilon'\sqrt{k}m$, then by Theorem 3 in polynomial time we can find $S, T \subseteq V$ such that

$$\left|R^{(i)}(S,T)\right| \geq \epsilon' \cdot \sqrt{k} \cdot m/2 \geq \epsilon' \cdot \sqrt{h(B)} \cdot m/2. \tag{2}$$

Choose $W^{(i+1)} = \mathrm{CUT}(S,T,\alpha)$, for $\alpha = R^{(i)}(S,T)/m^2$, and let $R^{(i+1)} = R^{(i)} - W^{(i+1)}$. We get,

$$h(R^{(i+1)}) - h(R^{(i)}) = -2\alpha R^{(i)}(S,T) + \alpha^2 d(S)d(T) \leq -\frac{R^{(i)}(S,T)^2}{m^2} \leq -\frac{\epsilon'^2 \cdot h(B)}{4}.$$

Since $h(R^{(0)}) = h(B)$, after $\sigma \leq 4/\epsilon'^2 = 16k/\epsilon^2$, we have $\left\|R^{(\sigma)}\right\|_C \leq \epsilon'\sqrt{k}m$, which implies that

$$\left\|A - W^{(1)} - \ldots - W^{(\sigma)}\right\|_C \leq \|A-B\|_C + \left\|B - W^{(1)} - \ldots - W^{(\sigma)}\right\|_C \leq \epsilon m.$$

This proves the correctness of the algorithm. It remains to upper bound α. For each cut matrix $W^{(i)} = \mathrm{CUT}(S,T,\alpha)$ constructed throughout the algorithm we have

$$\begin{aligned} |\alpha| = \frac{|R^{(i)}(S,T)|}{m^2} &= \frac{1}{m^2}\left|\sum_{u\in S, v\in T} R^{(i)}_{u,v}\frac{\sqrt{d(u)d(v)}}{\sqrt{d(u)d(v)}}\right| \\ &\leq \frac{1}{m^2}\sqrt{\sum_{u\in S, v\in T}\frac{{R^{(i)}_{u,v}}^2}{d(u)d(v)}}\sqrt{d(S)d(T)} \\ &\leq \frac{\sqrt{h(R^{(i)})}}{m} \leq \frac{\sqrt{h(B)}}{m} = \frac{\sqrt{k}}{m}. \end{aligned} \tag{3}$$

where the first inequality follows by the Cauchy-Schwarz inequality, the second inequality uses $d(S), d(T) \leq m$, and the last inequality follows by the fact that the potential function is decreasing throughout the algorithm. This completes the proof of theorem. □

4 Fast Approximation Algorithm for Low Threshold Rank Graphs

In this section we prove Theorem 2. First, by Theorem1 in time $\text{poly}(n, 1/\epsilon)$ we can find cut matrices $W^{(1)}, \ldots, W^{(\sigma)}$ for $\sigma = O(k/\epsilon^2)$, such that for all $1 \leq i \leq t$, $W^{(i)} = \text{CUT}(S_i, T_i, \alpha_i)$, $\alpha_i \leq \sqrt{k}/m$, and

$$\|A - W\|_C \leq \epsilon m/4,$$

where $W := W^{(1)} + \ldots + W^{(\sigma)}$. It follows from the above equation that for any set $S \subseteq V$,

$$|A(S, \overline{S}) - W(S, \overline{S})| = |A(S, \overline{S}) - \sum_{i=1}^{\sigma} \alpha_i \cdot d(S \cap S_i) \cdot d(\overline{S} \cap T_i)| \leq \frac{\epsilon m}{4}. \quad (4)$$

Fix $S^* \subseteq V$ of size $d(S^*) = \Gamma$ (think of $(S^*, \overline{S^*})$ as the optimum cut), and let $s_i^* := d(S_i \cap S^*)$, and $t_i^* := d(T_i \cap \overline{S^*})$. Observe that by equation (4),

$$\left| A(S^*, \overline{S^*}) - \sum_{i=1}^{\sigma} \alpha_i s_i^* t_i^* \right| \leq \frac{\epsilon m}{4}. \quad (5)$$

Let $\alpha_{\max} := \max_{1 \leq i \leq \sigma} |\alpha_i|$. Choose $\Delta = \Theta(\epsilon^3 m/k^{1.5})$ such that

$$\Delta \leq \min\left\{ \frac{\epsilon^3 \cdot m}{48}, \frac{\epsilon}{48 \alpha_{\max} \cdot \sigma} \right\}. \quad (6)$$

Note that this is achievable since $k \geq 1$.

We define an approximation of s_i^*, t_i^* by rounding them down to the nearest multiple of Δ, i.e., $\tilde{s}_i^* = \Delta \lfloor s_i^*/\Delta \rfloor$, and $\tilde{t}_i^* = \Delta \lfloor t_i^*/\Delta \rfloor$. We use $\tilde{s}^*, \tilde{t}^*$ to denote the vectors of the approximate values. It follows that we can obtain a good approximation of the size of the cut $(S^*, \overline{S^*})$ just by guessing the vectors $\tilde{s}^*$, and $\tilde{t}^*$. Since $|s_i^* - \tilde{s}_i^*| \leq \Delta$ and $|t_i^* - \tilde{t}_i^*| \leq \Delta$, we get,

$$\sum_{i=1}^{\sigma} |s_i^* t_i^* \alpha_i - \tilde{s}_i^* \tilde{t}_i^* \alpha_i| \leq \sigma \cdot \alpha_{\max}(2 \cdot \Delta \cdot m + \Delta^2) \leq 3 \alpha_{\max} \cdot \sigma \cdot \Delta \cdot m \leq \epsilon \cdot m/16. \quad (7)$$

where we used (6).

Observe that by equations (4),(5),(7), if we know the vectors $\tilde{s}^*, \tilde{t}^*$, then we can find $A(S^*, \overline{S^*})$ within an additive error of $\epsilon m/2$. Since $\tilde{s}_i^*, \tilde{t}_i^* \leq m$, there are only $O(m/\Delta)$ possibilities for each $\tilde{s}_i^*$ and $\tilde{t}_i^*$. Therefore, we afford to enumerate all possible values of them in time $(m/\Delta)^{2\sigma}$, and choose the one that gives the largest cut. Unfortunately, for a given assignment of $\tilde{s}^*, \tilde{t}^*$ the corresponding cut $(S^*, \overline{S^*})$ may not exist. Next we give an algorithm that for a given assignment of $\tilde{s}^*, \tilde{t}^*$ finds a cut $(S, \overline{S})$ such that $A(S, \overline{S}) = \sum_i \tilde{s}_i^* \tilde{t}_i^* \alpha_i \pm \epsilon m$, if one exists.

First we distinguish the large degree vertices of G and simply guess which side they are mapped to in the optimum cut. For the rest of the vertices we use the

solution of LP(1). Let $U := \{v : d(v) \geq \Delta\}$ be the set of large degree vertices. Observe that $|U| \leq m/\Delta$. Let $\mathcal{P}$ be the coarsest partition of the set $V \setminus U$ such that for any $1 \leq i \leq \sigma$, both $S_i \setminus U$ and $T_i \setminus U$ can be written as a union of sets in $\mathcal{P}$, and for each $P \in \mathcal{P}$, $d(P) \leq \Delta$. Observe that $|\mathcal{P}| \leq 2^{2\sigma} + m/\Delta$. For a given assignment of $\tilde{s}^*, \tilde{t}^*$, first we guess the set of vertices in U that are contained in S^*, $U_{S^*} := S^* \cap U$, and $U_{\overline{S^*}} := U \setminus U_{S^*}$. For the rest of the vertices we use the linear program LP(1) to find the unknown $d(S^* \cap P)$.

LP(1)

$$\begin{aligned}
0 &\leq y_P \leq 1 && \forall P \in \mathcal{P} \\
\Gamma - \epsilon m/2 &\leq \sum_P y_P d(P) + d(U_{S^*}) \leq \Gamma + \epsilon m/2 && (8) \\
\tilde{s}_i^* &\leq \sum_{P \subseteq S_i} y_P d(P) + d(U_{S^*} \cap S_i) \leq \tilde{s}_i^* + \Delta && \forall 1 \leq i \leq \sigma \\
& && (9) \\
\tilde{t}_i^* &\leq \sum_{P \subseteq T_i} (1 - y_P) d(P) + d(U_{\overline{S^*}} \cap T_i) \leq \tilde{t}_i^* + \Delta && \forall 1 \leq i \leq \sigma. \\
& && (10)
\end{aligned}$$

Observe that $y_P = \frac{d(S^* \cap P)}{d(P)}$ is a feasible solution to the linear program. In the next lemma which is the main technical part of the analysis we show how to construct a set based on a given solution of the LP.

Lemma 3. *There is a randomized algorithm such that for any $S^* \subset V$, given $\tilde{s}_i^*, \tilde{t}_i^*$ and U_{S^*} returns a random set S such that*

$$\mathbb{P}\left[W(S, \overline{S}) \geq A(S^*, \overline{S^*}) - \frac{3\epsilon m}{4} \wedge |d(S) - \Gamma| \leq \epsilon m\right] \geq \frac{\epsilon}{10} \tag{11}$$

$$\mathbb{P}\left[W(S, \overline{S}) \leq A(S^*, \overline{S^*}) + \frac{3\epsilon m}{4} \wedge |d(S) - \Gamma| \leq \epsilon m\right] \geq \frac{\epsilon}{10}. \tag{12}$$

Proof. Let y be a feasible solution of LP(1). We use a simple independent rounding scheme to compute the random set S. We always include U_{S^*} in S. For each $P \in \mathcal{P}$, we include P in S, independently, with probability y_P. We prove that S satisfies lemma's statements. First of all, by linearity of expectation,

$$\mathbb{E}\left[d(S \cap S_i)\right] = d(U_{S^*}) + \sum_{P \subseteq S_i} y_P d(P), \quad \text{and}$$

$$\mathbb{E}\left[d(\overline{S} \cap T_i)\right] = d(U_{\overline{S^*}}) + \sum_{P \subseteq T_i} (1 - y_P) d(P).$$

In the following two claims, first we show that with high probability the size of $d(S)$ is close to Γ. Then, we upper bound the expected value of $W(S, \overline{S}) - A(S^*, \overline{S^*})$.

Claim 1

$$\mathbb{P}\left[|d(S) - d(S^*)| \geq \epsilon m\right] \leq \frac{\epsilon}{8},$$

Proof. We use the theorem of Hoeffding to prove the claim:

Theorem 4 (Hoeffding Inequality). *Let $X_1, \ldots, X_n$ be independent random variables such that for each $1 \leq i \leq n$, $X_i \in [0, a_i]$. Let $X := \sum_{i=1}^{n} X_i$. Then, for any $\epsilon > 0$*

$$\mathbb{P}\left[|X - \mathbb{E}\left[X\right]| \geq \epsilon\right] \leq 2\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{n} a_i^2}\right).$$

Now, by the independent rounding procedure, we obtain

$$\begin{aligned}\mathbb{P}\left[|d(S) - \mathbb{E}\left[d(S)\right]| \geq \epsilon m/2\right] &\leq 2\exp\left(-\frac{\epsilon^2 m^2}{2\sum_P d(P)^2}\right) \leq 2\exp\left(-\frac{\epsilon^2 m^2}{2m\Delta}\right) \\ &\leq 2\exp(-16/\epsilon) \leq \frac{\epsilon}{8}.\end{aligned}$$

where the second inequality follows by the fact that $d(P) \leq \Delta$ and $\sum_P d(P) \leq m$ and the third inequality follows by (6). The claim follows from the fact that by (8), $|\mathbb{E}\left[d(S)\right] - \Gamma| \leq \epsilon m/2$. □

Claim 2

$$\left|\mathbb{E}\left[W(S, \overline{S})\right] - A(S^*, \overline{S^*})\right| \leq \frac{\epsilon m}{2}.$$

Proof. First, observe that

$$\begin{aligned}\mathbb{E}\left[W(S, \overline{S})\right] &= \mathbb{E}\left[\sum_{i=1}^{\sigma} d(S \cap S_i) d(\overline{S} \cap T_i)\alpha_i\right] \\ &= \sum_{i=1}^{\sigma} \alpha_i \mathbb{E}\left[\left(\sum_{P \in \mathcal{P}: P \subseteq S_i} d(P)\mathbb{I}\left[P \subseteq S\right]\right)\left(\sum_{Q \in \mathcal{P}: Q \subseteq T_i} d(Q)\mathbb{I}\left[Q \subseteq \overline{S}\right]\right)\right] \\ &\quad + \sum_{i=1}^{\sigma} \alpha_i \left(d(U_{S^*} \cap S_i)\mathbb{E}\left[d(\overline{S} \cap T_i)\right] + d(U_{\overline{S^*}} \cap T_i)\mathbb{E}\left[d(S \cap S_i)\right]\right) \\ &= \sum_{i=1}^{\sigma} \alpha_i \sum_{P \subseteq S_i, Q \subseteq T_i} d(P)d(Q)\mathbb{E}\left[\mathbb{I}\left[P \subseteq S\right]\mathbb{I}\left[Q \subseteq \overline{S}\right]\right] \\ &\quad + \sum_{i=1}^{\sigma} \alpha_i \left(d(U_{S^*} \cap S_i)t_i + d(U_{\overline{S^*}} \cap T_i)s_i\right). \qquad (13)\end{aligned}$$

where $s_i := \mathbb{E}\left[d(S \cap S_i)\right]$ and $t_i := \mathbb{E}\left[d(\overline{S} \cap T_i)\right]$.

Since the event that $P \subseteq S$ is independent of $Q \subseteq S$, iff $P \neq Q$ we get

$$\mathbb{E}\left[\mathbb{I}\left[P \subseteq S\right]\mathbb{I}\left[Q \subseteq \overline{S}\right]\right] = \begin{cases} y_P(1 - y_Q) & \text{if} P \neq Q \\ 0 & \text{otherwise.} \end{cases}$$

Then, by (13) and above equation,

$$\mathbb{E}\left[W(S,\overline{S})\right] = \sum_{i=1}^{\sigma} \alpha_i s_i t_i - \sum_{i=1}^{\sigma} \alpha_i \sum_{P\in\mathcal{P}} y_P(1-y_P)d(P)^2. \tag{14}$$

On the other hand, by equations (9) and (10), for all $1 \le i \le \sigma$, we get $\tilde{s}_i^* \le s_i \le \tilde{s}_i^* + \Delta$ and $\tilde{t}_i^* \le t_i \le \tilde{t}_i^* + \Delta$. Hence, similar to equation (7) we can show,

$$\sum_{i=1}^{\sigma} |\alpha_i s_i t_i - \alpha_i \tilde{s}_i^* \tilde{t}_i^*| \le \frac{\epsilon m}{8}. \tag{15}$$

Therefore, using equation (5) we get

$$\begin{aligned}
&\left|\mathbb{E}\left[W(S,\overline{S})\right] - A(S^*,\overline{S^*})\right| \\
&\qquad \le \left|\mathbb{E}\left[W(S,\overline{S})\right] - \sum_{i=1}^{\sigma} \alpha_i s_i^* t_i^*)\right| + \frac{\epsilon m}{4} \\
&\qquad = \left|\sum_{i=1}^{\sigma} (\alpha_i s_i t_i - \alpha_i s_i^* t_i^*) - \sum_{i=1}^{\sigma}\sum_{P\in\mathcal{P}} \alpha_i y_P(1-y_P)d(P)^2\right| + \frac{\epsilon m}{4} \\
&\qquad \le \sum_{i=1}^{\sigma} |\alpha_i s_i t_i - \alpha_i \tilde{s}_i^* \tilde{t}_i^*| + \sum_{i=1}^{\sigma} |\alpha_i \tilde{s}_i^* \tilde{t}_i^* - \alpha_i s_i^* t_i^*| + \sigma \alpha_{\max} m\Delta + \frac{\epsilon m}{4} \\
&\qquad \le \frac{\epsilon m}{2},
\end{aligned}$$

where the equality follows by (14), the second inequality follows by the fact that $d(P) \le \Delta$ for all $P \in \mathcal{P}$ and $\sum_P d(P) \le m$, and the last inequality follows by equations (15) and (7). This proves the claim. □

Now we are ready finish the proof of Lemma 3. Here, we prove (11). Equation (12) can be proved similarly. By Claim 2,

$$\begin{aligned}
A(S^*,\overline{S^*}) - \frac{\epsilon m}{2} &\le \mathbb{E}\left[W(S,\overline{S})\right] \\
&\le \mathbb{E}\left[W(S,\overline{S}) \mid |d(S) - \Gamma| \le \epsilon m\right] + m\mathbb{P}\left[|d(S) - \Gamma| > \epsilon m\right] \\
&\le \mathbb{E}\left[W(S,\overline{S}) \mid |d(S) - \Gamma| \le \epsilon m\right] + \frac{\epsilon m}{8}.
\end{aligned}$$

where the second inequality holds by the fact that the size of any cut in G is at most $m/2$, thus by (4) for any $S \subseteq V$, $W(S,\overline{S}) \le \epsilon m/4 + m/2 \le m$, and the last inequality follows by Claim 1. Hence,

$$\mathbb{E}\left[W(S,\overline{S}) \mid |d(S) - \Gamma| \le \epsilon m\right] \ge A(S^*,\overline{S^*}) - \frac{5\epsilon m}{8}$$

Since $W(S,\overline{S}) \le m$,

$$\mathbb{P}\left[W(S,\overline{S}) \ge A(S^*,\overline{S^*}) - \frac{3\epsilon m}{4} \;\middle|\; |d(S) - \Gamma| \le \epsilon m\right] \ge \frac{\epsilon}{8}$$

Therefore, (11) follows by an application of Claim 1. □

Our rounding algorithm is described in Algorithm 1. First, we prove the correctness, then we calculate the running time of the algorithm. Let S be the output set of the algorithm. First, observe that the output always satisfy $|d(S) - \Gamma| \leq \epsilon m$. Now let $A(S^*, \overline{S^*})$ be the maximum cut among all sets of size Γ (the minimization case can be proved similarly). In the iteration that the algorithm correctly guesses $\tilde{s}_i^*, \tilde{t}_i^*, U_{S^*}$, there exists a feasible solution y of LP(1). by Lemma 3, for all $1 \leq i \leq 10/\epsilon$,

$$\mathbb{P}\left[W(R_y(i), \overline{R_y(i)}) \geq A(S^*, \overline{S^*}) - \frac{3\epsilon m}{4} \wedge |d(R_y(i)) - \Gamma| \leq \epsilon m\right] \geq \frac{\epsilon}{10}$$

Since we take the best of $10/\epsilon$ samples, with probability $1/e$ the output set S satisfies $W(S, \overline{S}) \geq A(S^*, \overline{S^*}) - 3\epsilon m/4$. Therefore, by (4), $A(S, \overline{S}) \geq A(S^*, \overline{S^*}) - \epsilon m$. This proves the correctness of the algorithm.

Algorithm 1. Approximate Maximum Cut $(S, \overline{S})$ such that $d(S) = \Gamma \pm \epsilon m$

for all possible values of $\tilde{s}_i^*, \tilde{t}_i^*$, and $U_{S^*} \subseteq U$ **do**
  if there is a feasible solution y of LP(1) **then**
    for $i = 1 \to 10/\epsilon$ **do**
      $R_y(i) \leftarrow U_{S^*}$.
      For each $P \in \mathcal{P}$ include P in $R_y(i)$, independently, with probability y_P.
    end for
  end if
end for
return among all sets $R_y(i)$ sampled in the loop that satisfy $|d(R_y(i)) - \Gamma| \leq \epsilon m$, the one that $W(R_y(i), \overline{R_y(i)})$ is the maximum.

It remains to upper-bound the running time of the algorithm. First observe that if $|U| = O(k/\epsilon^2)$, the running time of the algorithm is dominated by the time it takes to compute a feasible solution of LP(1). Since the size of LP is $2^{\tilde{O}(k/\epsilon^2)}$, in this case Algorithm 1 terminates in time $2^{\tilde{O}(k/\epsilon^2)}$. Note that for any sample set $R_y(i)$, both $d(R_y(i))$ and $W(R_y(i), \overline{R_y(i)})$ can be computed in time $2^{\tilde{O}(k/\epsilon^2)}$, once we know $|R_y(i) \cap P|$ for any $P \in \mathcal{P}$.

Otherwise if $|U| \gg k/\epsilon^2$, the dependency of the running time of the algorithm to ϵ, k is dominated by the step where we guess the subset of $U_{S^*} = U \cap S^*$. Since $\alpha_{\max} \leq \sqrt{k}/m$ and $\sigma = O(k/\epsilon^2)$, we get

$$|U| \leq \frac{m}{\Delta} \leq \frac{12m\alpha_{\max}\sigma}{\epsilon} = O\left(\frac{k^{1.5}}{\epsilon^3}\right).$$

Therefore, Algorithm 1 runs in time $2^{\tilde{O}(k^{1.5}/\epsilon^3)}$. Since it takes $\text{poly}(n, k, 1/\epsilon)$ to compute the decomposition into $W^{(1)}, \ldots, W^{(\sigma)}$, the the total running time is $2^{\tilde{O}(k^{1.5}/\epsilon^3)} + \text{poly}(n)$. This completes the proof of Theorem 2.

Acknowledgement. We would like to thank anonymous reviewer for helpful comments.

References

[ABS10] Arora, S., Barak, B., Steurer, D.: Subexponential algorithms for unique games and related problems. In: FOCS, pp. 563–572 (2010)

[AN06] Alon, N., Naor, A.: Approximating the cut-norm via grothendieck's inequality. SIAM J. Comput. 35(4), 787–803 (2006)

[BRS11] Barak, B., Raghavendra, P., Steurer, D.: Rounding semidefinite programming hierarchies via global correlation. In: FOCS, pp. 472–481 (2011)

[CCF09] Coja-Oghlan, A., Cooper, C., Frieze, A.: An efficient sparse regularity concept. In: SODA, pp. 207–216. Society for Industrial and Applied Mathematics, Philadelphia (2009)

[DKKV05] Fernandez De la Vega, W., Karpinski, M., Kannan, R., Vempala, S.: Tensor decomposition and approximation schemes for constraint satisfaction problems. In: STOC, pp. 747–754. ACM, New York (2005)

[FK96] Frieze, A.M., Kannan, R.: The regularity lemma and approximation schemes for dense problems. In: FOCS, p. 12. IEEE Computer Society, Washington, DC (1996)

[FK99] Frieze, A.M., Kannan, R.: Quick approximation to matrices and applications. Combinatorica 19(2), 175–220 (1999)

[GS11] Guruswami, V., Sinop, A.K.: Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with psd objectives. In: FOCS, pp. 482–491 (2011)

[GS12] Guruswami, V., Sinop, A.K.: Faster sdp hierarchy solvers for local rounding algorithms. In: FOCS (2012)

[GS13] Guruswami, V., Sinop, A.K.: Lasserre sdps, l_1-embeddings, and approximating non-uniform sparsest cut via generalized spectra. In: SODA (2013)

[GW95] Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM 42(6), 1115–1145 (1995)

[Kol11] Kolla, A.: Spectral algorithms for unique games. Comput. Complex. 20(2), 177–206 (2011)

[KT07] Kolla, A., Tulsiani, M.: Playing unique games using graph spectra. Manuscript (2007)

[Sze78] Szemeredi, E.: Regular partitions of graphs. In: Problemes Combinatoires et Theorie des Graphes. Colloq. Internat. CNRS, vol. 260, pp. 399–401. CNRS, Paris (1978)

Interdiction Problems on Planar Graphs

Feng Pan[1,⋆] and Aaron Schild[2,⋆⋆,⋆⋆⋆,†]

[1] D-6, Los Alamos National Laboratory
fpan@lanl.gov
[2] Princeton University
aschild@princeton.edu

Abstract. We introduce approximation algorithms and strong NP-completeness results for interdiction problems on planar graphs. Interdiction problems are leader-follower games in which the leader is allowed to delete a certain number of edges from the graph in order to maximally impede the follower, who is trying to solve an optimization problem on the impeded graph. We give a multiplicative $(1+\epsilon)$-approximation algorithm for the weighted maximum matching interdiction problem on weighted planar graphs. The algorithm runs in pseudo-polynomial time for each fixed $\epsilon > 0$. We also show that weighted maximum matching interdiction remains strongly NP-complete on planar graphs. In the process, we show that the budget-constrained flow improvement, directed shortest path interdiction, and minimum perfect matching interdiction problems are strongly NP-complete on planar graphs. To our knowledge, our budget-constrained flow improvement result is the first planar NP-completeness proof that uses a one-vertex crossing gadget.

Keywords: maximum matching, maximum flow, interdiction, planar graphs, crossing gadget, approximation scheme.

1 Introduction

Interdiction problems are often used to understand the robustness of solutions to combinatorial optimization problems on graphs. For any optimization problem

⋆ Supported by Defense Threat Reduction Agency grant BRCALL08-A-2-0030.

⋆⋆ Supported by the Office of the Dean of the College, Princeton University, the Center for Nonlinear Studies at Los Alamos National Laboratory, the DHS-STEM Research Fellowship Program, and NSF grant CCF-0964037.

⋆⋆⋆ This research was performed under an appointment to the U.S. Department of Homeland Security (DHS) Science and Technology (S&T) Directorate Office of University Programs HS-STEM Summer Internship Program, administered by the Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy (DOE) and DHS. ORISE is managed by the Oak Ridge Associated Universities (ORAU) under DOE contract number DE-AC05-06OR23100. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, and ORAU/ORISE.

† Work done partially while visting Los Alamos National Laboratory.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 317–331, 2013.

on a graph, one can formulate an interdiction variant by creating a leader-follower game. In edge interdiction problems, every edge of the graph has an interdiction cost associated with it. The leader is given a budget and is allowed to delete any set of edges with total cost less than the given budget. The follower then solves the given optimization problem on the remaining graph. The leader wants to pick the set of edges to delete that impedes the follower as much as possible.

In this paper, we focus on edge interdiction problems relating to shortest path interdiction, maximum flow interdiction, and matching interdiction. We give formal definitions for these problems here.

For a given directed graph G with a nonnegative edge capacities, let $\alpha(G, s, t)$ denote the value of the maximum flow from s to t. The Budget-Constrained Flow Improvement Problem (BCFIP) [20], a problem that is closely related to the maximum flow interdiction problem, is defined as follows:

Input: A directed graph $G = (V, E)$ with a capacity function $w : E \to \mathbb{Z}_{\geq 0}$, a transport cost function $c : E \to \mathbb{Z}_{\geq 0}$, an integer budget $B > 0$, and two distinct distinguished nodes $s, t \in V$.
Output: A subset $N \subseteq E$ with $c(N) \leq B$ that maximizes $\alpha(G[N], s, t)$, where $G[N]$ denotes the subgraph of G induced by the edges in N.

Let $\alpha_B^E(G, s, t)$ denote the optimal value for BCFIP.

Let $\rho(G, u, v)$ denote the total weight of the shortest path between u and v in an edge-weighted graph G. The Directed Shortest Path Edge Interdiction Problem (DSPEIP) is defined as follows:

Input: An edge-weighted directed graph $G = (V, E)$ with a weight function $w : E \to \mathbb{Z}_{\geq 0}$, an interdiction cost function $c : E \to \mathbb{Z}_{\geq 0}$, an integer interdiction budget $B > 0$, and two distinct nodes $u, v \in V$.
Output: A subset $I \subseteq E$ with $c(I) \leq B$ that maximizes $\rho(G \backslash I, u, v)$.

Let $\rho_B^E(G, u, v)$ denote the optimal value for DSPEIP.

A *perfect matching* in a graph is a matching such that every vertex in the graph is incident with some edge in the matching. For an edge-weighted graph G, let $\mu(G)$ denote the weight of the minimum weight perfect matching if it exists. If no perfect matching exists in G, let $\mu(G) = \infty$. An instance of the Minimum Perfect Matching Edge Interdiction Problem (MPMEIP) is specified as follows:

Input: An edge-weighted graph $G = (V, E)$ with edge weight function $w : E \to \mathbb{Z}_{\geq 0}$, interdiction cost function $c : E \to \mathbb{Z}_{\geq 0}$, and interdiction budget $B > 0$. It is assumed that, for every set $I \subseteq E$ with $c(I) \leq B$, $G \backslash I$ has a perfect matching.
Output: A subset $I \subseteq E$ with $c(I) \leq B$ that maximizes $\mu(G \backslash I)$.

Let $\mu_B^E(G)$ denote the optimal value for MPMEIP.

A *matching* in a graph is a set of edges such that no two edges share an endpoint. For an edge-weighted graph $G = (V, E)$ with edge weight function

$w : E \to \mathbb{Z}_{\geq 0}$, let $\nu(G)$ be the weight of a maximum weight matching in G. The Maximum Matching Edge Interdiction Problem (MMEIP), originally defined by Zenklusen in [23], is defined as follows:

Input: An edge-weighted graph $G = (V, E)$ with a weight function $w : E \to \mathbb{Z}_{\geq 0}$, an interdiction cost function $c : E \to \mathbb{Z}_{\geq 0}$, and an integer interdiction budget $B > 0$.

Output: A subset $I \subseteq E$ with $c(I) \leq B$ that minimizes $\nu(G \backslash I)$.

Let $\nu_B^E(G)$ denote the optimal value for MMEIP.

1.1 Prior Work

Interdiction problems have practical applications in assessing the robustness of infrustructure networks. Some previous applications include drug trafficking [22], military planning [9], protecting utility networks from terrorist attacks [14], and controlling the spread of an infection [2].

Many researchers have worked towards understanding the complexity of interdiction problems on general graphs. BCFIP is known to be strongly NP-complete on bipartite graphs and weakly NP-complete on series parallel graphs [20]. Directed shortest path interdiction is strongly NP-complete on general graphs [16] and is strongly NP-hard to approximate within any factor better than 2 on general graphs [15]. A node-wise variant of the shortest path interdiction problem is solvable in polynomial time [15]. Heuristic solutions are known for shortest path interdiction. Israeli and Wood [13] gave a MIP formulation and used Benders' Decomposition to solve it efficiently on graphs with fewer than 5000 vertices.

The maximum flow interdiction problem is strongly NP-hard on general graphs, though a continuous variant of the problem has a pseudoapproximation that for every $\epsilon > 0$ either returns a solution within a $1 + \frac{1}{\epsilon}$-factor of the optimum or returns a better than optimal solution that uses at most $1 + \epsilon$ times the allocated budget [5, 22]. A standard linear programming relaxation of the maximum flow interdiction problem, even after adding two families of valid inequalities, has an integrality gap of $\Omega(n^{1-\epsilon})$ for all $\epsilon \in (0, 1)$ [1].

MMEIP is strongly NP-complete on bipartite graphs, even with unit edge weights and interdiction costs. Zenklusen [23] introduced a constant factor approximation algorithm for MMEIP on graphs with unit edge weights. This algorithm makes use of iterative LP rounding. Dinitz and Gupta provided a constant-factor approximation algorithm for a generalization of matching interdiction called *packing interdiction*[6]. Zenklusen [23] also showed that MMEIP is solvable in pseudo-polynomial time on graphs with bounded treewidth.

Fewer researchers have worked on interdiction problems restricted to planar graphs. Phillips [18] gave a pseudo-polynomial time algorithm for the directed maximum flow edge interdiction problem. Zenklusen [24] extended this algorithm to allow for some vertex deletions and showed that the planar densest k-subgraph problem reduces to maximum flow interdiction with multiple sources and sinks on planar graphs. Zenklusen [23] left the complexity of matching interdiction on planar graphs as an open problem.

1.2 Our Contributions

In this paper, we give a pseudo-polynomial time approximation scheme for MMEIP on undirected planar graphs. A *pseudo-polynomial time approximation scheme* (Pseudo-PTAS) is an algorithm that takes a parameter $\epsilon \in (0, 1)$ as additional input and outputs a solution with objective value within a multiplicative $(1 + \epsilon)$-factor of the optimum (or a $(1 - \epsilon)$ factor for maximization problems). Furthermore, the algorithm terminates in pseudo-polynomial time for fixed ϵ. In Section 3, we give an algorithm that achieves the following guarantee:

Theorem 1. *Let I be an edge set with $c(I) \leq B$ that minimizes $\nu(G \backslash I)$. There is an algorithm that, for every $\epsilon > 0$, returns a set $\widehat{I}$ of edges for which*

$$\nu(G \backslash \widehat{I}) \leq (1 + \epsilon)\nu(G \backslash I)$$

and $c(\hat{I}) \leq B$. Furthermore, for every fixed ϵ, the algorithm runs in polynomial time with respect to the size of the graph, the sum of the edge weights, and the sum of the interdiction costs of all edges (pseudo-polynomial time).

We also show that BCFIP, DSPEIP, MPMEIP, and MMEIP are strongly NP-complete even on planar graphs. The proofs of Theorems 3, 4, and 5 are deferred to the full version of the paper [17], while the proof of Theorem 2 is given in Section 4.

Theorem 2. *Given an integer budget $B > 0$, an integer $k \geq 0$, a directed planar graph G with polynomially-bounded edge integer capacities and edge transport costs ($w(e), c(e) \leq P(|V(G)|)$ for all $e \in E(G)$ and a fixed polynomial $P : \mathbb{Z} \to \mathbb{Z}$), and two vertices s and t adjacent to a common face, it is NP-complete to decide whether or not $\alpha_B^E(G, s, t) > k$.*

Theorem 3. *Given an edge-weighted directed planar graph G with polynomially-bounded integer edge weights and interdiction costs, an integer $B > 0$, an integer $k > 0$, and two vertices $u, v \in V(G)$, it is NP-complete to decide whether or not $\rho_B^E(G, u, v) > k$.*

Theorem 4. *Given an edge-weighted undirected bipartite planar graph G with polynomially-bounded integer edge weights and interdiction costs, an integer $B > 0$, and an integer $k > 0$, it is NP-complete to decide whether or not $\mu_B^E(G) > k$.*

Theorem 5. *Given an edge-weighted undirected bipartite planar graph G with polynomially-bounded integer edge weights and interdiction costs, an integer $B > 0$, and an integer $k > 0$, it is NP-complete to decide whether or not $\nu_B^E(G) < k$.*

The strong NP-completeness of MMEIP on planar graphs implies that our Pseudo-PTAS is optimal with respect to approximation ratio, resolving the question asked by Zenklusen [23] about the complexity of MMEIP on planar graphs.

To obtain our Pseudo-PTAS for MMEIP, we extend Baker's technique for interdiction problems on planar graphs. We give two simple conditions for local maximization problems on graphs that guarantee the existence of a (Pseudo-)PTAS for their interdiction variants. Let ζ be a real-valued function on the set of undirected graphs with the following two properties:

1. (P1) For any subgraph $H \subseteq G$, $\zeta(H) \leq \zeta(G)$.
2. (P2) For a partition of the edge set $E_1 \cup E_2 = E$ of G, $\zeta(G[E_1]) + \zeta(G[E_2]) \geq \zeta(G)$, where $G[E_1]$ denotes the subgraph of G induced by the edges in E_1.

Consider an interdiction cost function $c : E \to \mathbb{Z}_{\geq 0}$. If a set $I \subseteq E$ with $c(I) \leq B$ that minimizes $\zeta(G \backslash I)$ (the interdiction variant of ζ) can be computed in (pseudo-)polynomial time for graphs G with bounded treewidth, then we show that there is a (Pseudo-)PTAS for the interdiction variant of ζ on planar graphs. Maximum weight matchings satisfy these conditions, which we show gives a Pseudo-PTAS for MMEIP.

We show that the BCFIP problem is strongly NP-complete on directed planar graphs with a novel crossing removal technique. We start by offering a new method for showing that BCFIP is strongly NP-complete on general graphs. This method uses a reduction from the maximum independent set problem. We note that if no two crossing edges have the same capacity, then we can add one vertex at every crossing without changing the value of the maximum flow in which every edge has no flow or full flow. We exploit the observation by introducing a sweepline technique that assigns weights and costs to the edges in order to ensure that an optimal BCFIP flow must use full flow or no flow at all edges. Our NP-completeness result differs from prior NP-completeness results on planar graphs (e.g. [8, 7, 21]) in that it uses the specific structure of instances arising from the maximum independent set problem in order to replace crossings with just one vertex.

We defer complexity results depending on the strong NP-completeness of planar BCFIP to the full version [17]. First, we show that the reduction from maximum flow interdiction (with source and sink on the same face) to multi-objective shortest path on planar graphs [18] also gives an approximation-preserving reduction from planar BCFIP to planar DSPEIP. A well-known reduction from the shortest path problem to the assignment problem [11] fails to preserve planarity. We give a new approximation-preserving and planarity-preserving reduction that reduces DSPEIP to MPMEIP. We also reduce MPMEIP to MMEIP on planar graphs using edge weight manipulations. Our results distinguish BCFIP, DSPEIP, MPMEIP, and MMEIP from the maximum flow interdiction problem, which is solvable in pseudo-polynomial time on planar graphs.

In Section 2, we give some notation that we use throughout this paper. In Section 3, we present our Pseudo-PTAS for MMEIP. In Section 4, we prove that BCFIP is strongly NP-complete on planar graphs. We conclude this extended abstract with Section 5, in which we discuss open problems relating to interdiction.

2 Preliminaries

For a (undirected or directed) graph G, let $V(G)$ denote its vertex set and $E(G)$ denote its edge set. Edges in a directed graph are denoted by ordered pairs (u, v) for $u, v \in V(G)$, while edges in an undirected graph are denoted

Problem name	General hardness	Planar hardness
Max-Flow Interdiction	Strongly NP-C	Pseudo-polynomial[18]
DSPEIP	$(2-\epsilon)$-inapproximable (NP-H) [15]	**Strongly NP-C**
BCFIP	Strongly NP-C [20]	**Strongly NP-C**
MPMEIP (**introduced**)	**Strongly NP-C**	**Strongly NP-C**
MMEIP	Strongly NP-C [23]	**Strongly NP-C**

Problem name	General approximation	Planar approximation
Max-Flow Interdiction	Open	1[18]
DSPEIP	Open	Open
BCFIP	Open	Open
MPMEIP (**introduced**)	Open	Open
MMEIP	O(1) weighted [6], 4 unweighted[23]	**Pseudo-PTAS**

Fig. 1. A summary of known results about the problems we consider. Our results are displayed in bold. "Open" means that no nontrivial approximation algorithm is known. "NP-C" abbreviates "NP-complete." "NP-H" abbreviates "NP-hard."

by unordered pairs $\{u, v\}$. For a (undirected or directed) graph G, let $G[U]$ denote the subgraph induced by the vertices in $U \subseteq V(G)$, i.e. the subgraph of G for which $V(G[U]) = U$ and $E(G[U]) = \{\{u, v\} \in E(G) : u, v \in U\}$ (undirected graph, for a directed graph replace $\{u, v\}$ with (u, v)). For $F \subseteq E(G)$, let $G[F]$ denote the subgraph of G induced by the edges in F, i.e. the graph with $V(G[F]) = \{v \in V(G) : \exists w\{w, v\} \in F\}$ (undirected) or $V(G[F]) = \{v \in V(G) : \exists w(w, v) \in F \vee (v, w) \in F\}$ (directed) and $E(G[F]) = F$.

For a set $S \subseteq V(G)$, let $\delta(S) \subseteq E(G)$ be the set of edges with exactly one endpoint in S. For a directed graph, let $\delta^+(S) = \{(u, v) \in E(G) : u \in S, v \notin S\}$ and let $\delta^-(S) = \{(u, v) \in E(G) : u \notin S, v \in S\}$. Note that $\delta(S) = \delta^-(S) \cup \delta^+(S)$ for any $S \subseteq V(G)$ if G is directed. For a vertex $v \in V(G)$, let $\delta(v) = \delta(\{v\})$, $\delta^+(v) = \delta^+(\{v\})$, and $\delta^-(v) = \delta^-(\{v\})$. For any function $f : E(G) \to \mathbb{R}$ and $F \subseteq E(G)$, let $f(F) = \sum_{e \in F} f(e)$.

For a set S, let 2^S denote its power set. For an undirected graph G, a tree decomposition [19, 10] of G is a pair $(T, f : V(T) \to 2^{V(G)})$ where T is a tree. Furthermore, f has the following properties:

- $\cup_{w \in V(T)} f(w) = V(G)$
- For all $\{u, v\} \in V(G)$, there is some $w \in V(T)$ such that $u, v \in f(w)$.
- For any $v \in V(G)$, let $U_v \subseteq V(T)$ be the set of vertices $w \in V(T)$ for which $v \in f(w)$. Then, $T[U_v]$ is connected.

Let $k_T = \max_{w \in V(T)}(|f(w)| - 1)$ denote the *width* of T. The treewidth of G is the minimum width of any tree decomposition of G.

For an integer $k > 0$, call an undirected graph G a k-outerplanar graph if it is planar and every vertex of G is at most $k - 1$ edges away from a distinguished face called the infinite face. Bodlaender [4] showed the following theorem:

Theorem 6. *k-outerplanar graphs have treewidth at most $3k-1$. Furthermore, a tree decomposition of width at most $3k-1$ can be found in $O(kn)$ time.*

3 A Pseudo-PTAS for Maximum Matching Interdiction

In this section, we introduce a pseudo-polynomial time approximation scheme for the maximum matching interdiction problem (MMEIP). We will use many ideas from Baker's framework introduced in [3]. First, we will state the following three properties of maximum weight matchings, which encapsulate our use of the structure of matchings to obtain a Pseudo-PTAS:

Proposition 1. *Let G be an edge-weighted graph with weight function $w : E(G) \to \mathbb{Z}_{\geq 0}$ and let H be a subgraph of G. Then, $\nu(H) \leq \nu(G)$.*

Proposition 2. *Let G be an edge-weighted graph with weight function $w : E(G) \to \mathbb{Z}_{\geq 0}$. Consider any $F \subseteq E(G)$. Let G_1 be the subgraph of G induced by the edge set F and let G_2 be the subgraph of G induced by the edge set $E(G) \backslash F$. Then, $\nu(G) \leq \nu(G_1) + \nu(G_2)$.*

Proposition 3. *Let G be an edge-weighted graph with weight function $w : E(G) \to \mathbb{Z}_{\geq 0}$. Pick any node $v \in V(G)$ and do a breadth-first search starting from v and mark the vertices of G with their distance from v. Let E_i be the set of edges from a vertex marked with i to a vertex marked with $i+1$. Fix an integer $k > 1$. For all $i \in \{0, 1, \ldots, k-1\}$, let $F_i = \cup_{j=0}^{\lfloor \frac{r-i}{k} \rfloor} E_{i+jk}$ (where r is the radius of the graph) and let G_i be the subgraph of G induced by F_i. Then,*

$$\sum_{i=0}^{k-1} \nu(G_i) \leq 2\nu(G)$$

Proof. Note that for $k > 1$, no edge in E_{i+jk} shares an endpoint with any edge in $E_{i+j'k}$ for $j \neq j'$. Therefore, if $\widehat{G}_i$ denotes the subgraph of G induced by E_i, then

$$\nu(G_i) = \sum_{j=0}^{\lfloor \frac{r-i}{k} \rfloor} \nu(\widehat{G}_{i+jk})$$

Let $F' = \cup_{j=0}^{\lfloor \frac{r}{2} \rfloor} E_{2j}$ and let $F'' = \cup_{j=0}^{\lfloor \frac{r-1}{2} \rfloor} E_{2j+1}$. Let G' be the subgraph of G induced by F' and let G'' be the subgraph of G induced by F''. Then, reindexing shows that

$$\sum_{i=0}^{k-1} \nu(G_i) = \sum_{j=0}^{r} \nu(\widehat{G}_j) = \nu(G') + \nu(G'') \leq 2\nu(G)$$

Now, we will use these properties to construct a Pseudo-PTAS. We will use Zenklusen's algorithm for bounded treewidth graphs in [23] as a subroutine. Consider the following algorithm:

Data: An edge-weighted graph G with edge weight function $w : E(G) \to \mathbb{Z}_{\geq 0}$, interdiction cost function $c : E(G) \to \mathbb{Z}_{\geq 0}$, interdiction budget $B \geq 0$, and an approximation threshold $\epsilon > 0$
Result: An interdiction set $\widehat{I} \subseteq E(G)$ with $c(\widehat{I}) \leq B$
Let $k = \lceil \frac{2}{\epsilon} \rceil$;
Fix an arbitrary vertex v;
Do a BFS of G from v to obtain edge sets $\{E_i\}_{i=0}^{r}$;
foreach $i \in \{0, 1, \ldots, k-1\}$ **do**
  Let $F_i = \cup_{j=0}^{\lfloor \frac{r-i}{k} \rfloor} E_{i+jk}$;
  Let G_i be the subgraph of G induced by $E(G) \backslash F_i$;
  Let H_i be the subgraph of G induced by F_i;
  for $b = 0$, $b \leq B$, $b = b + 1$ **do**
    Run Zenklusen's Algorithm on G_i with budget b to obtain a set $I_{ib1} \subseteq E(G_i)$;
    Run Zenklusen's Algorithm on H_i with budget $B - b$ to obtain a set $I_{ib2} \subseteq E(H_i)$;
    Let $I_{ib} = I_{ib1} \cup I_{ib2}$;
  end
end
Return the set I_{ib} for $i \in \{0, 1, \ldots, k-1\}$ and $b \in \{0, 1, \ldots, B\}$ that minimizes $\nu(G \backslash I_{ib})$;

Algorithm 1. Algorithm for MMEIP on planar graphs

We prove Theorem 1 by proving two theorems about the performance of Algorithm 1. We restate them here:

Theorem 7 (Approximation guarantee for Algorithm 1). *Let I be an edge set with $c(I) \leq B$ that minimizes $\nu(G \backslash I)$. The set $\widehat{I}$ returned by Algorithm 1 satisfies*

$$\nu(G \backslash \widehat{I}) \leq (1 + \epsilon)\nu(G \backslash I)$$

Proof. By Proposition 3 applied to $G \backslash I$, there is some i such that $\nu(H_i \backslash I) \leq \frac{2}{k}\nu(G \backslash I)$. Let $b = c(I \cap (E(G) \backslash F_i))$. Since $c(I) \leq B$, $c(I \cap F_i) \leq B - b$. Zenklusen's algorithm returns the optimal interdiction sets on G_i and H_i for the budgets b and $B - b$ respectively. Therefore, $\nu(G_i \backslash I_{ib1}) \leq \nu(G_i \backslash I)$ and $\nu(H_i \backslash I_{ib2}) \leq \nu(H_i \backslash I)$. By Proposition 1, $\nu(G_i \backslash I) \leq \nu(G \backslash I)$. Summing inequalities and applying Proposition 2 shows that

$$\begin{aligned} \nu(G \backslash I_{ib}) &\leq \nu(G_i \backslash I_{ib1}) + \nu(H_i \backslash I_{ib2}) \\ &\leq \nu(G_i \backslash I) + \nu(H_i \backslash I) \\ &\leq (1 + \frac{2}{k})\nu(G \backslash I) \\ &\leq (1 + \epsilon)\nu(G \backslash I) \end{aligned}$$

Since $\nu(G \backslash \widehat{I}) \leq \nu(G \backslash I_{ib})$, we are done.

Theorem 8 (Runtime guarantee for Algorithm 1). *For fixed ϵ, this algorithm terminates in pseudo-polynomial time on planar graphs. More precisely, on planar graphs, it terminates in time*

$$O((2B/\epsilon)(|V(G)|(C+1)^{8^{2/\epsilon+2}}) + (2B/\epsilon)|E(G)|\sqrt{|V(G)|})$$

where $C = w(E(G))$.

Proof. Zenklusen's algorithm [23] has runtime $O(|V(G)|(C+1)^{2^{t+1}})$ on graphs with treewidth at most t. The breadth-first search at the beginning of the algorithm takes $O(|E| + |V|)$ time. The innermost for loop is run $B+1$ times per execution of the outermost for loop, which runs at most $k \leq \frac{2}{\epsilon} + 1$ times. By Theorem 6, the treewidths of G_i and H_i are at most $3k-1$ and 2 respectively. The computation on G_i dominates the computation on H_i. Using the Hopcroft-Karp algorithm [12] to find the best interdiction set on the last line of Algorithm 1 gives the last term of the runtime.

4 Strong NP-Hardness of Budget-Constrained Flow Improvement on Planar Graphs

In this section, we sketch the proof of Theorem 2. We reduce from the maximum independent set problem on general graphs. Before outlining the proof Theorem 2, we discuss the proof of the following easier result, which was shown using a different reduction in [20]:

Lemma 1. *The decision version of BCFIP is strongly NP-complete on general directed graphs.*

Proof. Since maximum flow is in P, BCFIP is in NP. To show NP-hardness, we reduce from the maximum independent set problem on general graphs. Consider an undirected graph G and suppose we are given an integer $k > 0$ for which we want to determine if the maximum independent set in G contains more than k vertices. Create a graph G_1 with one vertex for every vertex of G, one vertex for every edge of G, and two distinguished vertices s and t. The edges in G_1 are split into four sets E_1, E_2, E_3, and E_4. There are directed edges in G_1 of the following types:

1. One edge from s to each vertex corresponding to a vertex of G. (E_1)
2. For every edge $e = \{u, v\} \in E(G)$, two directed edges (u, e) and (v, e) in $E(G_1)$. (E_2)
3. One edge from each vertex corresponding to an edge of G to t. (E_3)
4. One edge from each vertex corresponding to a vertex of G to t. (E_4)

Let d be the maximum degree of any node in G. The capacity function $w_1 : E(G_1) \to \mathbb{Z}_{\geq 0}$ is:

$$w_1(e) = \begin{cases} d & : e \in E_1 \\ 1 & : e \in E_2 \\ 1 & : e \in E_3 \\ d - \deg_G(\eta(e)) & : e \in E_4 \end{cases}$$

where $\eta : E_4 \to V(G)$ returns the vertex of G corresponding to the left endpoint of the input edge of G_1. The cost function $c_1 : E(G_1) \to \mathbb{Z}_{\geq 0}$ is:

$$c_1(e) = \begin{cases} 1 : e \in E_1 \\ 0 : e \notin E_1 \end{cases}$$

This construction is depicted in Figure 2. The proof of the following proposition is deferred to the full version of the paper [17].

Proposition 4. *$\alpha_k^E(G_1, s, t) = kd$ if and only if there is an independent set in G of size at least k.*

We prove Theorem 2 by removing crossings from G_1. First, we will informally discuss the key idea. Note that the existence of an independent set of size at least k guaranteed that all edges transfered either no flow or full flow. There is only enough budget to pay for edges with full flow in the network. This means that if two edges with different edge weights cross and we add a node at the intersection of the two edges, there is no way for the flow to "change direction," as doing so would leave an edge with partial flow. Since there is not enough money to pay for edges with partial flow, we ensure that adding a vertex at the crossing does not change the behavior of the network.

We now describe how to remove crossings from G_1 to obtain a graph G_2 with an associated capacity function $w_2 : E(G_2) \to \mathbb{Z}_{\geq 0}$ and cost function $c_2 : E(G_2) \to \mathbb{Z}_{\geq 0}$. Every edge in G_2 is a segment between two edge crossings in G_1 and originates from a *parent edge* in G_1. The transport cost of an edge is $w_e(s_e + 1)$, where w_e is the weight and s_e is the number of sweeplines that e crosses (not including the endpoints):

1. (Sweepline creation step) Embed G_1 in an $x - y$ coordinate system so that it has the following properties:
   - s has coordinates $(0, 0)$
   - t has coordinates $(1, 0)$
   - all right endpoints of edges in E_1 (left endpoints of E_2 and E_4) have x-coordinate $\frac{1}{3}$
   - all right endpoints of edges in E_2 (left endpoints of E_3) have x-coordinate $\frac{2}{3}$
   - all edges in E_1, E_2, and E_3 are embedded as line segments
   - all edges in E_4 are embedded so that they do not cross edges in E_3
2. (Crossing addition step) Obtain G_2 by adding vertices at all edge crossings in the $x-y$ coordinate embedding of G_1. Note that all of the added crossings will have x-coordinate in the interval $(\frac{1}{3}, \frac{2}{3})$. These added vertices split edges of E_2 and E_3 into *child edges*. An edge in G_2 has a unique *parent edge* in G_1 from which it was obtained by vertex additions at crossings.

3. (Edge weighting step) Now, construct w_2 as follows:
   (a) Arbitrarily label the vertices of G_1 that correspond to edges of G with the integers 1 through $|E(G)|$ inclusive.
   (b) For any edge $e \in E(G_2)$ with its parent (a copy of itself) in E_1 (within G_1), let $w_2(e) = |E(G)|^2$.
   (c) For any edge $e \in E(G_2)$ with its parent in E_2, let $w_2(e)$ be the label of the right endpoint of the parent of e in G_1.
   (d) For any edge $e \in E(G_2)$ with its parent (a copy of itself) in E_3, let $w_2(e)$ be the label of the left endpoint of the parent of e in G_1.
   (e) For any edge $\{u, v\} \in E(G_2)$ with its parent in E_4, let

$$w_2(\{u, v\}) = |E(G)|^2 - \sum_{e \in \delta^+(u) \cap \kappa(\delta^+(u) \cap E_2)} w_2(e)$$

   where $\kappa : 2^{E(G_1)} \to 2^{E(G_2)}$ returns the set of all child edges for a given input set of edges in G_1.
4. (Transport cost step) Finally, construct c_2 as follows:
   (a) Sort the crossing vertices added to G_1 in increasing order by x-coordinate (with ties broken arbitrarily) to obtain a list $\{v_i\}_{i=1}^r$, where r denotes the number of added points.
   (b) For any edge $e \in E(G_2)$ with its parent in E_1, let $c_2(e) = w_2(e) = |E(G)|^2$.
   (c) For any edge $e = \{v_i, v_j\} \in E(G_2)$ with its parent in E_2 or E_4, let $c_2(e) = (j - i)w_2(e)$.
   (d) For any edge $e = \{u, v_i\} \in E(G_2)$ with its parent in E_2 or E_4 and u having x-coordinate $\frac{1}{3}$, let $c_2(e) = iw_2(e)$.
   (e) For any edge $e = \{v_i, w\} \in E(G_2)$ with its parent in E_2 and w having x-coordinate $\frac{2}{3}$, let $c_2(e) = (r + 1 - i)w_2(e)$.
   (f) For any edge $e = \{v_i, t\} \in E(G_2)$ with its parent in E_4, let $c_2(e) = (r + 2 - i)w_2(e)$.
   (g) For any edge $e = \{u, w\} \in E(G_2)$ with its parent in E_2, u with x-coordinate $\frac{1}{3}$, and w with x-coordinate $\frac{2}{3}$, let $c_2(e) = (r + 1)w_2(e)$.
   (h) For any edge $e = \{u, t\} \in E(G_2)$ with its parent in E_4 and u with x-coordinate $\frac{1}{3}$, let $c_2(e) = (r + 2)w_2(e)$.
   (i) For any edge $e = \{w, t\} \in E(G_2)$ with its parent in E_3 and w with x-coordinate $\frac{2}{3}$, let $c_2(e) = w_2(e)$.

Suppose that G_1 had r edge crossings. Note that $r \leq \binom{|E(G)|}{2} + |V(G)||E(G)|$, which is polynomially bounded in the size of the graph. This crossing removal construction depicted in Figure 3 has several properties, which follow from the three properties previously stated:

Proposition 5. *For any positive integer k, the total edge cost of a flow with value at least $k|E(G)|^2$ must be at least $(r + 3)k|E(G)|^2$.*

Proof. The embedding construction in Part 2 of the construction can be associated with $r+3$ cuts in order by x-coordinate. The cost of an edge e in this network is $w_2(e)$ times the number of these $r+3$ cuts that e is in. Therefore, it suffices to show that at least $k|E(G)|^2$ units of cost are required to send $k|E(G)|^2$ units of flow accross the cut. This follows from the definition of the cost function, completing this proof.

Proposition 6. *For any positive integer k, the total edge cost of a flow with value at least $k|E(G)|^2$ that has partial flow along at least one edge must be at least $(r+3)k|E(G)|^2+1$.*

Proof. Let $e \in E(G_2)$ be an edge with partial flow under a flow $f : E(G_2) \to \mathbb{Z}_{\geq 0}$. Consider one of the $r+3$ cuts discussed in the proof of Lemma 5 that contains e. It suffices to show that the cost of edges in this cut is at least $k|E(G)|^2+1$. Since the capacities and flows through edges are integers, $c_2(e)-f(e) \geq 1$, which implies the result.

Proposition 7. *Let G_2 be the planar graph that results from edge crossing removal. Consider two edges $e, e' \in E(G_1)$ that cross in the $x-y$ embedding of G_1. Consider any edges $f, f' \in E(G_2)$ that originated from e and e' respectively after crossing removal. Then, $w_2(f) \neq w_2(f')$.*

Proof. Note that all children of e have the same maximum capacity. The same holds for children of e'. Parts (a) and (c) of Part 3 of the construction ensure that if $e, e' \in E_2$, then $w_2(f) \neq w_2(f')$. Note that no two edges in E_4 cross in the embedding of G_1. Therefore, the only other possible scenario with a crossing occurs when $e \in E_2$ and $e' \in E_4$. It suffices to show that $w_2(f') \geq \frac{|E(G)|^2-|E(G)|}{2} > |E(G)|$ if $|E(G)| \geq 4$ (if $|E(G)| \leq 3$, we can solve the instance with brute force in constant time). This suffices because the weight of any edge with a parent in E_2 is at most $|E(G)|$.

Since children of e' have the same capacity, we may let f' be the unique child that has as a left endpoint e''s left endpoint. Let w be this left endpoint. Since G_1 is a simple graph, w has at most $|E|+1$ outgoing edges and the sum of the capacities of edges besides f' must be at most $\sum_{n=1}^{|E|} = \frac{|E|^2+|E|}{2}$. By Part (3e),

$$w_2(f') = |E(G)|^2 - \sum_{e \in \delta^+(u) \cap \kappa(\delta^+(u) \cap E_2)} w_2(e) \geq \frac{|E(G)|^2 - |E(G)|}{2}$$

as desired.

These three properties are important for proving the next lemma. This lemma implies Theorem 2 since the reduction takes polynomial time, the edge weights are polynomially bounded in the size of the graph, and G_2 is planar. We defer the proof of the lemma to the full version of the paper [17].

Lemma 2. *G has an independent set of size at least k if and only if there is a flow on G_2 with total edge cost at most $(r+3)k|E(G)|^2$ with value $k|E(G)|^2$.*

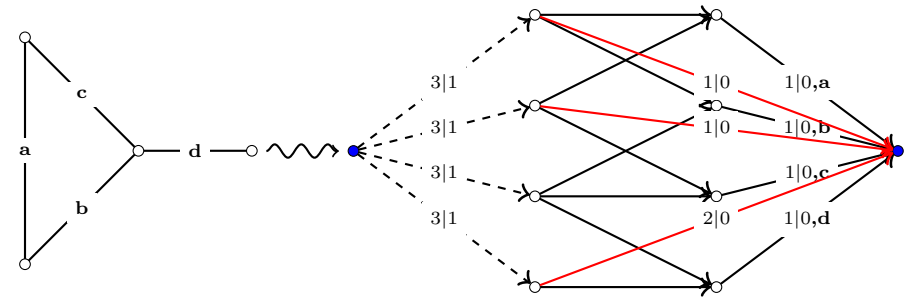


Fig. 2. The reduction in the proof of Lemma 1, with $w|c$ as the edge labels. Dashed edges must be paid for, while dark edges are never deleted (since they have cost 0). All edges in E_2 (which are the unlabeled edges here) have label 1|0.

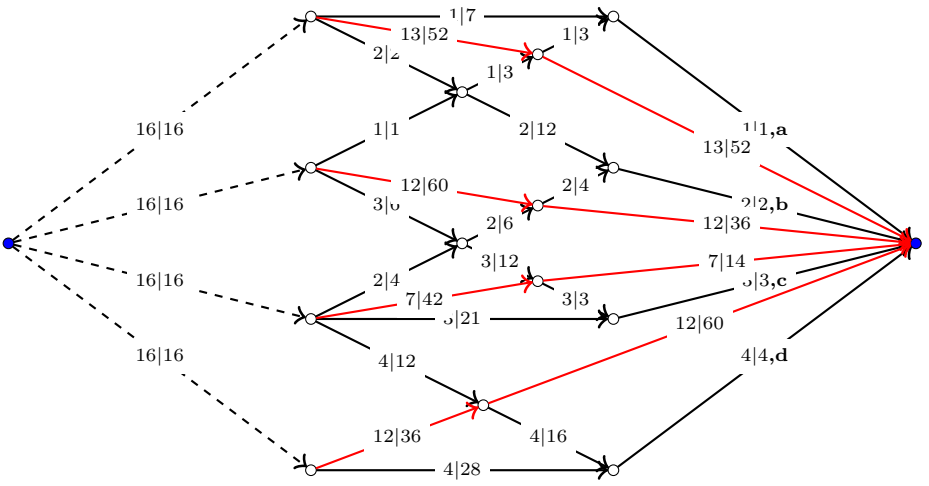


Fig. 3. The result of crossing removal in the proof of Theorem 2 on the second graph in Figure 2. Labels are $w|c$. Note that six crossings were added, so $r = 6$ in this example. It costs at least 9 units of budget to send a unit of flow from s to t. Furthermore, at any added crossing, the capacities of the incoming edges are distinct.

5 Conclusion and Open Problems

In this extended abstract, we described the complexity of edge interdiction problems when restricted to planar graphs. We presented a Pseudo-PTAS for the weighted maximum matching interdiction problem (MMEIP) on planar graphs. The algorithm extends Baker's Technique for local bilevel min-max optimization problems on planar graphs. Furthermore, we gave strong NP-hardness results for budget-constrained maximum flow (BCFIP), directed shortest path interdiction, minimum perfect matching interdiction, and maximum matching interdiction (MMEIP) on planar graphs. The latter three results followed from the strong

NP-completeness of BCFIP on directed planar graphs with source and sink on a common face. This strong NP-completeness proof first reduced the maximum independent set problem to BCFIP on general directed graphs. We then noticed that optimal flows on these directed graphs had either full flow or no flow along all edges. To take advantage of this fact, we introduced a sweepline technique for assigning transportation costs and edge capacities in order to ensure that no two crossing edges had the same capacity. After introducing this embedding technique, edge crossings could be replaced with just one vertex.

There are many interesting open problems relating to interdiction. While hardness of approximation results are known for shortest path interdiction [15], it is only known that a $(2 - \epsilon)$-factor pseudo-polynomial time approximation would imply that $P = NP$. While heuristic solutions are known for the shortest path interdiction problem [13], no approximation algorithms with nontrivial approximation (or pseudoapproximation) guarantees are known. For the maximum matching interdiction problem, Zenklusen [23] showed several hardness of approximation results. Nonetheless, no hardness of approximation results are known for MMEIP (computing $\nu_B^E(G)$ within a multiplicative factor). A pseudoapproximation is known for a continuous variant of the maximum flow interdiction problem [5]. However, no algorithms and no hardness of approximation results are known for the (discrete) maximum flow interdiction problem on general directed graphs.

While we now know that BCFIP, directed shortest path interdiction, mimimum perfect matching interdiction, and MMEIP are strongly NP-complete on planar graphs, there are no known Pseudo-PTASes for BCFIP, shortest path interdiction, and minimum perfect matching interdiction when restricted to planar graphs.

References

[1] Altner, D.S., Ergun, Ö., Uhan, N.A.: The maximum flow network interdiction problem: Valid inequalities, integrality gaps, and approximability. Oper. Res. Lett. 38(1), 33–38 (2010)

[2] Assimakopoulos, N.: A network interdiction model for hospital infection control. Comput. Biol. Med. 17(6), 413–422 (1987)

[3] Baker, B.S.: Approximation algorithms for np-complete problems on planar graphs. J. ACM 41(1), 153–180 (1994)

[4] Bodlaender, H.L.: Some classes of graphs with bounded treewidth. Bulletin of the EATCS 36, 116–125 (1988)

[5] Burch, C., Carr, R., Krumke, S., Marathe, M., Phillips, C., Sundberg, E.: A decomposition-based pseudoapproximation algorithm for network flow inhibition. In: Woodruff, D.L. (ed.) Network Interdiction and Stochastic Integer Programming. Operations Research/Computer Science Interfaces Series, vol. 22, pp. 51–68. Springer, US (2003)

[6] Dinitz, M., Gupta, A.: Packing interdiction and partial covering problems. In: Goemans, M., Correa, J. (eds.) IPCO 2013. LNCS, vol. 7801, pp. 157–168. Springer, Heidelberg (2013)

[7] Garey, M.R., Johnson, D.S.: The rectilinear steiner tree problem in np complete. SIAM Journal of Applied Mathematics 32, 826–834 (1977)
[8] Garey, M.R., Johnson, D.S., Tarjan, R.E.: The planar hamiltonian circuit problem is np-complete. SIAM J. Comput. 5(4), 704–714 (1976)
[9] Montgomery, D.C., Ghare, P.M., Turner, W.C.: Optimal interdiction policy for a flow network. Naval Research Logistics Quarterly 18, 37–45
[10] Halin, R.: S-functions for graphs. Journal of Geometry 8(1-2), 171–186 (1976)
[11] Hoffman, A.J., Markowitz, H.M.: A note on shortest path, assignment, and transportation problems. Naval Research Logistics Quarterly 10(1), 375–379 (1963)
[12] Hopcroft, J.E., Karp, R.M.: A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. In: SWAT (FOCS), pp. 122–125 (1971)
[13] Israeli, E., Kevin Wood, R.: Shortest-path network interdiction. Networks 40, 2002 (2002)
[14] Kevin Wood, R., Salmeron, J., Baldick, R.: Worst-case interdiction analysis of large-scale electric power grids. IEEE Transactions on Power Systems 24, 96–104 (2009)
[15] Khachiyan, L., Boros, E., Borys, K., Elbassioni, K.M., Gurvich, V., Rudolf, G., Zhao, J.: On short paths interdiction problems: Total and node-wise limited interdiction. Theory Comput. Syst. 43(2), 204–233 (2008)
[16] Malik, K., Mittal, A.K., Gupta, S.K.: The k most vital arcs in the shortest path problem. Oper. Res. Lett. 8(4), 223–227 (1989)
[17] Pan, F., Schild, A.: Interdiction problems on planar graphs (2013), `http://arxiv.org/abs/1305.1407`
[18] Phillips, C.A.: The network inhibition problem. In: STOC, pp. 776–785 (1993)
[19] Robertson, N., Seymour, P.D.: Graph minors. iii. Planar tree-width. Journal of Combinatorial Theory, Series B 36(1), 49–64 (1984)
[20] Schwarz, S., Krumke, S.: On budget-constrained flow improvement. Information Processing Letters 66(6), 291–297 (1998)
[21] Stockmeyer, L.: Planar 3-colorability is polynomial complete. SIGACT News 5(3), 19–25 (1973)
[22] Kevin Wood, R.: Deterministic network interdiction. Mathematical and Computer Modelling 17(2), 1–18 (1993)
[23] Zenklusen, R.: Matching interdiction. Discrete Applied Mathematics 158(15), 1676–1690 (2010)
[24] Zenklusen, R.: Network flow interdiction on planar graphs. Discrete Applied Mathematics 158(13), 1441–1455 (2010)

Conditional Random Fields, Planted Constraint Satisfaction and Entropy Concentration

Emmanuel Abbe[1] and Andrea Montanari[2]

[1] Department of Electrical Engineering and Program in Applied and Computational Mathematics, Princeton University
eabbe@princeton.edu
[2] Departments of Electrical Engineering and Statistics, Stanford University
montanari@stanford.edu

Abstract. This paper studies a class of probabilistic models on graphs, where edge variables depend on incident node variables through a fixed probability kernel. The class includes planted constraint satisfaction problems (CSPs), as well as more general structures motivated by coding and community clustering problems. It is shown that under mild assumptions on the kernel and for sparse random graphs, the conditional entropy of the node variables given the edge variables concentrates around a deterministic threshold. This implies in particular the concentration of the number of solutions in a broad class of planted CSPs, the existence of a threshold function for the disassortative stochastic block model, and the proof of a conjecture on parity check codes. It also establishes new connections among coding, clustering and satisfiability.

Keywords: Planted models, constraint satisfaction problems, graphical models, community clustering, parity-check codes, entropy, concentration, interpolation method.

1 Introduction

This paper studies a class of probabilistic models on graphs encompassing models of statistical learning theory, coding theory, and random combinatorial optimization. Depending on the framework, the class may be described as a family of conditional random fields, memory channels, or planted constrained satisfaction problems. We start by providing motivations in the latter framework.

Constrained satisfaction problems (CSPs) are key components in the theory of computational complexity as well as important mathematical models in various applications of computer science, engineering and physics. In CSPs, a set of variables $x_1, \dots, x_n$ is required to satisfy a collection of constraints involving each a subset of the variables. In many cases of interest, the variables are Boolean and the constraints are all of a common type: e.g., in k-SAT, the constraints require the OR of k Boolean variables or their negations to be TRUE, whereas in k-XORSAT, the XOR of the variables or their negations must equal to zero. Given a set of constraints and a number of variables, the problem is to decide whether

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 332–346, 2013.

there exists a satisfying assignment. In random CSPs, the constraints are drawn at random from a given ensemble, keeping the constraint density[1] constant. In this setting, it is of interest to estimate the *probability* that a random instance is satisfiable. One of the fascinating phenomenon occurring for random instances is the phase transition, which makes the task of estimating this probability much easier in the limit of large n. For a large class of CSPs, and as n tends to infinity, the probability of being satisfiable tends to a step function, jumping from 1 to 0 when the constraint density crosses a critical threshold. For random k-XORSAT the existence of such a critical threshold is proved [18,17,15,41]. For random k-SAT, $k \geq 3$, the existence of a n-dependent threshold is proved in [24]. However it remains open to show that this threshold converges when n tends to infinity. Upper and lower bounds are known to match up to a term that is of relative order $k\,2^{-k}$ as k increases [9]. Phase transition phenomena in other types of CSPs are also investigated in [7,39,9]

In planted random CSPs, a "planted assignment" is first drawn, and the constraints are then drawn at random so as to keep that planted assignment a satisfying one. Planted ensembles were investigated in [11,28,6,5,29,3], and at high density in [8,4,22]. In the planted setting, the probability of being SAT is always one by construction, and a more relevant question is to determine the actual *number* of satisfying assignments. One would expect that this problem becomes easier in the limit of large n due to an asymptotic phenomenon. This paper shows that, indeed, a concentration phenomenon occurs: for a large class of planted CSPs (including SAT, NAE-SAT and XOR-SAT) the normalized logarithm of the number or satisfying assignment concentrates around a deterministic number. Moreover, this deterministic threshold is n-independent.

It is worth comparing the result obtained in this paper for planted CSPs, with the one obtained in [1] for non planted CSPs. In that case, the number of solution is zero with positive probability and therefore the logarithm of the number of solution does not have a finite expectation. Technically, standard martingale methods does not allow to prove concentration, even around an n-dependent threshold. In [1] an interpolation method [26] is used to prove the existence of the limit of a 'regularized' quantity, namely the logarithm of the number of solutions plus one, divided by the number of variables. A technical consequence of this approach is that the concentration of this quantity around a value that is independent of n can only be proved when the UNSAT probability is known to be $O(1/\log(n)^{1+\varepsilon})$.

This paper shows that –in the planted case– the concentration around an n-independent value holds unconditionally and is exponentially fast. We use again the interpolation technique [26,20,21,42,10,1] but with an interesting twist. While in all the cited references, the entropy (or log-partition function) is shown to be superadditive, in the present setting it turns out to be subadditive.

Let us also mention that a fruitful line of work has addressed the relation between planted random CSPs and their non planted counterparts in the

[1] The ratio of the expected number of constraints per variables.

satisfiable phase [3], and in [32,45]. These papers show that, when the number of solutions is sufficiently concentrated, planting does not play a critical role in the model. It would be interesting to use these ideas to 'export' the concentration result obtained here to non planted models.

In this paper, we pursue a different type of approach. Motivated by applications[2], in particular in coding theory and community clustering, we consider extensions of the standard planted CSPs to a setting allowing soft probabilistic constraints. Within this setting, the planted solution is an unknown vector to be reconstructed, and the constraints are regarded as noisy observations of this unknown vector. For instance one can recover the case of planted random k-SAT as follows. Each clause is generated by selecting first k variable indices $i_1, \dots, i_k$ uniformly at random, representing the hyperedge of a random graph. Then a clause is drawn uniformly among the ones that are satisfied by the variables $x_{i_1}, \dots, x_{i_k}$ appearing in the planted assignment. The clause can hence be regarded as a noisy observation of $x_{i_1}, \dots, x_{i_k}$. More generally the formula can be seen as a noisy observation of the planted assignment.

Our framework extends the above to include numerous examples from coding theory and statistics. Within LDPC or LDGM codes [25,44], encoding is performed by evaluating the sum of a random subset of information bits and transmitting it through a noisy communication channel. The selection of the information bits is described by a graphs, drawn at random for the code construction, and the transmission of these bits leads to a noisy observation of the graph variables. Similarly, a community clustering block model [27] can be seen as a random graph model, whereby each edge is a noisy observation of the community assignments of the adjacent nodes. Definitions will be made precise in the next section.

The conditional probability of the unknown vector given the noisy observations takes the form of a graphical model, i.e. factorizes according to an hypergraph whose nodes correspond to variables and hyperedges correspond to noisy observations. Such graphical models have been studied by many authors in machine learning [33] under the name of 'conditional random fields', and in [38] in the context of LDPC and LDGM codes. The conditional entropy of the unknown vector given the observations is used here to quantify the residual uncertainty of the vector. This is equivalent to considering the mutual information between the node and edge variables. In such a general setting, we prove that the conditional entropy per variable concentrates around a well defined deterministic limit. This framework allows a unified treatment of a large class of interesting random combinatorial optimization problems, raises new connections among them, and opens up to new models. We obtain in particular a proof of a conjecture posed in [43] on low-density parity-check codes, and the existence of a threshold function for the disassortative stochastic block model [16].

[2] Planted models are also appealing to cryptographic application, as hard instances with known solutions provide good one-way functions.

2 The Model

Let k and n be two positive integers with $n \geq k$.

- Let $V = [n]$ and $g = (V, E(g))$ be a hypergraph with vertex set V and edge set $E(g) \subseteq E_k(V)$, where $E_k(V)$ denotes the set of all possible $\binom{n}{k}$ hyperedges of order k on the vertex set V. We will often drop the term "hyper".
- Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite sets called respectively the input and output alphabets. Let $Q(\cdot|\cdot)$ be a probability transition function (or channel) from $\mathcal{X}^k$ to $\mathcal{Y}$, i.e., for each $u \in \mathcal{X}^k$, $Q(\cdot|u)$ is a probability distribution on $\mathcal{Y}$.
- To each vertex in V, we assign a node-variable in $\mathcal{X}$, and to each edge in $E(g)$, we assign an edge-variable in $\mathcal{Y}$. We define

$$P_g(y|x) \equiv \prod_{I \in E(g)} Q(y_I|x[I]), \quad x \in \mathcal{X}^V, y \in \mathcal{Y}^{E(g)}, \tag{1}$$

  where y_I denotes the edge-variable attached to edge I, and $x[I]$ denotes the k node-variables attached to the vertices adjacent to edge I. This defines for a given hypergraph g the probability of the edge-variables given the node-variables.

The above is a type of factor or graphical model, or a planted constraint satisfaction problem with soft probabilistic constraints. For each $x \in \mathcal{X}^V$, $P_g(\cdot|x)$ is a product measure on the set of edge-variables. We call P_g a **graphical channel with graph g and kernel Q**. We next put the uniform probability distribution on the set of node-variables $\mathcal{X}^V$, and define the a posteriori probability distribution (or reverse channel) by

$$R_g(x|y) \equiv \frac{1}{S_g(y)} P_g(y|x) 2^{-n}, \quad x \in \mathcal{X}^V, y \in \mathcal{Y}^{E(g)}, \tag{2}$$

where

$$S_g(y) \equiv \sum_{x \in \mathcal{X}^V} P_g(y|x) 2^{-n} \tag{3}$$

is the output marginal distribution.

We now define two probability distributions on the hypergraph g, which are equivalent for the purpose of this paper:

- A sparse Erdös-Rényi distribution, where each edge is drawn independently with probability $p = \frac{\alpha n}{\binom{n}{k}}$, where $\alpha > 0$ is the edge density.
- A sparse Poisson distribution, where for each $I \in E_k(V)$, a number of edges m_I is drawn independently from a Poisson distribution of parameter $p = \frac{\alpha n}{\binom{n}{k}}$. Note that m_I takes value in $\mathbb{Z}$, hence G is now a multi-edge hypergraph. To cope with this more general setting, we allow the edge-variable y_I to take

value in $\mathcal{Y}^{m_I}$, i.e., $y_I = (y_I(1), \dots, y_I(m_I))$, and define (with a slight abuse of notation)

$$Q(y_I|x[I]) = \prod_{i=1}^{m_I} Q(y_I(i)|x[I]). \tag{4}$$

This means that for each I, m_I i.i.d. outputs are drawn from the kernel Q. If $m_I = 0$, no edge is drawn. We denote by $\mathcal{P}_k(\alpha, n)$ the above distribution on (multi-edge) hypergraphs.

Since $p = \frac{\alpha n}{\binom{n}{k}}$, the number of edges concentrates around its expectation given by αn and the two models are equivalent in the limit of large n — at least they are equivalent for the subsequent results.

3 Main Results

We now define the conditional entropy between the node and edge variables. This is equivalent, up to a constant shift, to the mutual information between the node and edge variables.

Definition 1. *Let X be uniformly drawn in $\mathcal{X}^n$, G be a random sparse hypergraph drawn from the $\mathcal{P}_k(\alpha, n)$ ensemble independently of X, and Y be the output of X through the graphical channel P_G defined in* (1) *for a kernel Q. We define*

$$H_G^{(n)}(X|Y) \equiv -2^{-n} \sum_{x \in \mathcal{X}^V} \sum_{y \in \mathcal{Y}^{E(G)}} P_G(y|x) \log R_G(x|y), \tag{5}$$

$$H^{(n)}(X|Y) \equiv \mathbb{E}_G H_G^{(n)}(X|Y), \tag{6}$$

where P_G and R_G are defined in (1) *and* (2) *respectively. Note that $H_G^{(n)}(X|Y)$ is a random variable since G is random, and for a realization $G = g$, $H_g^{(n)}(X|Y)$ is the conditional entropy of X given Y, which can be expressed as $H_g^{(n)}(X|Y) = \mathbb{E}_y H_g^{(n)}(X|Y = y)$. Note that the mutual information between the node and edge variables is also obtained as $I_g^{(n)}(X|Y) = n - H_g^{(n)}(X|Y)$.*

Definition 2. *We denote by $M_1(\mathcal{X}^l)$ the set of probability measures on $\mathcal{X}^l$. For a kernel Q from $\mathcal{X}^k$ to $\mathcal{Y}$, we define*

$$\Gamma_l : M_1(\mathcal{X}^l) \to \mathbb{R} \tag{7}$$

$$\nu \mapsto \Gamma_l(\nu) = \frac{1}{|\mathcal{Y}|} \sum_{u^{(1)}, \dots, u^{(l)} \in \mathcal{X}^k} \left[\sum_{z \in \mathcal{Y}} \prod_{r=1}^{l} (1 - Q(z|u^{(r)})) \right] \prod_{i=1}^{k} \nu(u_i^{(1)}, \dots, u_i^{(l)}) \,. \tag{8}$$

Hypothesis H. A kernel Q is said to satisfy hypothesis H if Γ_l is convex for any $l \geq 1$.

Despite the lengthy expression, it is important to note that the definition of Γ_l depends solely on the kernel Q. We will see in Section 4 that a large variety of kernels satisfy this hypothesis, including kernels corresponding to parity-check encoded channels, planted SAT, NAE-SAT, XORSAT, and disassortative stochastic block models.

We first show a sub-additivity property for the expected conditional entropy of graphical channels.

Theorem 1. *Let Q be a kernel satisfying hypothesis H and $n = n_1 + n_2$, with $n_1, n_2 \geq k$. Then*

$$H^{(n)}(X|Y) \leq H^{(n_1)}(X|Y) + H^{(n_2)}(X|Y). \tag{9}$$

The proof of this theorem is outlined in Section 5.

Corollary 1. *Let Q be a kernel satisfying hypothesis H. There exists $C_k(\alpha, Q)$ such that*

$$\frac{1}{n} H^{(n)}(X|Y) \to C_k(\alpha, Q), \quad \text{as } n \to \infty. \tag{10}$$

The following is obtained using previous corollary and a concentration argument.

Theorem 2. *Let Q be a kernel satisfying hypothesis H, then, almost surely,*

$$\lim_{n\to\infty} \frac{1}{n} H_G^{(n)}(X|Y) = C_k(\alpha, Q), \tag{11}$$

with $C_k(\alpha, Q)$ as in Corollary 1.

The proof can be found in [2].

4 Applications

We next present three applications of the general model described in previous section. While planted CSPs and parity-check codes are directly derived as particular cases of our model, the stochastic block model is obtained with a limiting argument. One of the advantages of relying on a general model class, is that it allows to consider new hybrid structures. For example, one may consider codes which are not linear but which rely on OR gates as in SAT, or on community models whose connectivity rely on collections of k nodes.

4.1 Planted Constraint Satisfaction Problems

Definition 3. *A CSP kernel is given by*

$$Q(z|u) = \frac{1}{|A(u)|} \mathbb{1}(z \in A(u)), \quad u \in \mathcal{X}^k, z \in \mathcal{Y}, \tag{12}$$

where $A(u)$ is a subset of $\mathcal{Y}$ containing the "authorized constraints", with the property that $|A(u)|$ is constant (it may depend on k but not on u).

We next show that a graphical channel with a CSP kernel corresponds to a planted CSP. We derive first a few known examples from this model.

- For planted k-SAT, $\mathcal{Y} = \{0,1\}^k$ and $A(u) = \{0,1\}^k \setminus \bar{u}$, where $\bar{u}$ is the vector obtained by flipping each component in u. Using this kernel for the graphical channel means that for any selected edge $I \in E_k(V)$, the edge variable y_I is a vector in $\{0,1\}^k \setminus \bar{x}[I]$ uniformly drawn, representing the negation pattern of the constraint I. Note that using u rather than $\bar{u}$ leads to an equivalent probabilistic model, $\bar{u}$ is simply used here to represent u as a "satisfying assignment". Note that $|A(u)| = 2^k - 1$.
- For planted k-NAE-SAT, $\mathcal{Y} = \{0,1\}^k$ and $A(u) = \{0,1\}^k \setminus \{u, \bar{u}\}$, with $|A(u)| = 2^k - 2$.
- For k-XOR-SAT, $\mathcal{Y} = \{0,1\}$ and $A(u) = \oplus_{i=1}^k u_i$ and $|A(u)| = 1$.

In general, a graphical channel with graph g and kernel Q as in (12) leads to a planted CSP where the constraints are given by $A(x[I]) \ni y_I$ for any $I \in E(g)$. For example, for planted k-SAT, the constraints are $\bar{x}[I] \neq y_I$, whereas for planted k-NAE-SAT, the constraints are $\bar{x}[I] \notin (y_I, \bar{y}_I)$. If y is drawn from the output marginal distribution S_g (cf. (3)), then there exists a satisfying assignment by construction.

Lemma 1. *For a graphical channel with graph g and CSP kernel Q as in* (12)*, and for y in the support of S_g,*

$$H_g(X|Y = y) = \log Z_g(y) \tag{13}$$

where $Z_g(y)$ is the number of satisfying assignments of the corresponding planted CSP.

Corollary 2. *For a graphical channel with CSP kernel Q as in* (12)*, and for a graph drawn from the ensemble $\mathcal{P}(\alpha, n)$,*

$$H^{(n)}(X|Y) = \mathbb{E}_{G,Y} \log Z_G(Y), \tag{14}$$

where $Z_G(Y)$ is the number of satisfying assignments of the corresponding random planted CSP.

Lemma 2. *For any $k \geq 1$, and for the CSP kernel corresponding to planted k-SAT, the operator Γ_l is convex for any $l \geq 1$.*

Lemma 3. *For any $k \geq 1$, and for the CSP kernel corresponding to planted k-NAE-SAT, the operator Γ_l is convex for any $l \geq 1$.*

Lemma 4. *For any k even, and for the CSP kernel corresponding to planted k-XOR-SAT, the operator Γ_l is convex for any $l \geq 1$.*

Using Theorem 2 and previous lemmas, the following is obtained.

Corollary 3. *For random planted k-SAT, k-NAE-SAT, and k-XOR-SAT (k even), the normalized logarithm of the number of solutions concentrates in probability.*

4.2 Stochastic Block Model

The problem of community clustering is to divide a set of vertices in a network (graph) into groups having a higher connectivity within the groups and lower connectivity across the groups (assortative case), or the other way around (disassortative case). This is a fundamental problem in many modern statistics, machine learning, and data mining problems with a broad range of applications in population genetics, image processing, biology and social science. A large variety of models have been proposed for community detection problems, we refer to [40,23,27] for a survey on the subject.

At an algorithmic level, the problem of finding the smallest cut in a graph with two equally sized groups, i.e., the min-bisection problem, is well-known to be NP-hard [14]. Concerning average-case complexity, various random graphs models have been proposed for community clustering. The Erdös-Rényi random graph is typically a very bad model for community structures, since each node is equally connected to any other nodes and no communities are typically formed. The stochastic block model is a natural extension of an Erdös-Rényi model with a community structure. Although the model is fairly simple (communities emerge but the average degree is still constant[3]), it is a fascinating model with several fundamental questions open.

We now describe the stochastic block model (SBM), also called planted bisection model, with two groups and symmetric parameters. Let $V = [n]$ be the vertex set and a, b be two positive real numbers. For a uniformly drawn assignment $X \in \{0,1\}^V$ on the vertices, an edge is drawn between vertex i and j with probability a/n if $X_i = X_j$ and with probability b/n if $X_i \neq X_j$, and each edge is drawn independently. We denote this model by $\mathcal{G}(n, a, b)$. Note that the average degree of an edge is $(a+b)/2$, however, a 0-labelled node is connected in expectation with $a/2$ 0-labeled nodes and with $b/2$ 1-labeled nodes.

This type of model was introduced in [14], in the dense regime. The attention to the sparse regime described above is more recent, with [13] and [31,16,36]. In particular, [16] conjectured a phase transition phenomenon, with the detection of clusters[4] being possible if $(a-b)^2 > 2(a+b)$ and impossible otherwise. In [36], a remarkable proof of the impossibility part is obtained, leaving the achievability part open.

We next define a parametrized kernel which will allow us to approximate the above SBM model with a graphical channel.

Definition 4. *An SBM kernel is given by*

$$Q(z|u_1,u_2) = \begin{cases} a/\gamma & \text{if } u_1 = u_2, \\ b/\gamma & \text{if } u_1 \neq u_2, \end{cases} \tag{15}$$

where $u_1, u_2, z \in \{0,1\}$.

[3] Models with corrected degrees have been proposed in [31].
[4] Obtaining a reconstruction positively correlated with the true assignment.

Lemma 5. *There exists $n_0 = n_0(\gamma, a, b)$ and $C = C(a, b)$ such that the following holds true. Let X be uniformly drawn on $\{0,1\}^V$, Y be the output (the graph) of a sparse stochastic block model of parameters a, b, and Y_γ be the output of a graphical channel with graph ensemble $\mathcal{P}(\gamma, n)$ and kernel* (15)*, then, for all $n \geq n_0$*

$$\left|H^{(n)}(X|Y) - H^{(n)}(X|Y_\gamma)\right| \leq \frac{Cn}{\gamma}. \tag{16}$$

Lemma 6. *For the SBM kernel given by* (15)*, $a \leq b$ (disassortative case) and γ large enough, the operator Γ_l is convex for any $l \geq 1$.*

Corollary 4. *For the disassortative SBM, the limit of $H(X^{(n)}|Y)/n$ exists and satisfies*

$$\lim_{n\to\infty} \frac{1}{n} H^{(n)}(X|Y) = \lim_{\gamma\to\infty} \lim_{n\to\infty} \frac{1}{n} H^{(n)}(X|Y_\gamma)\,. \tag{17}$$

In a work in progress, the assortative case is investigated with a different proof technique. The computation of the above limit is also expected to reflect a phase transition for the SBM [16,36].

4.3 Parity-Check Encoded Channels

The Shannon celebrated coding theorem states that for a discrete memoryless channel W from $\mathcal{X}$ to $\mathcal{Y}$, the largest rate at which reliable communication can take place is given by the capacity $C(W) = \max_X I(X;Y)$, where $I(X;Y)$ is the mutual information of the channel W with a random input X. To show that rates up to capacity are achievable, Shannon used random code books, relying on a probabilistic argument. Shortly after, Elias [19] showed that random linear codes allow to achieve capacity, reducing the encoding complexity from exponential to quadratic in the code dimension. However, Berlekamp, McEliece, and Van Tilborg showed in [12] that the maximum likelihood decoding of unstructured linear codes is NP-complete.

In order to reduce the complexity of the decoder, Gallager proposed the use sparse linear codes [25], giving birth to the LDPC codes, with sparse parity-check matrices, and LDGM codes, with sparse generator matrices. Various types of LDPC/LDGM codes depend on various types of row and column degree distributions. Perhaps one of the most basic class of such codes is the LDGM code with constant right degree, which corresponds to a generator matrix with column having a fixed number k of one's. This means that each codeword is the XOR of k uniformly selected information bits. In other words, this is a graph based code drawn from an Erdös-Rényi or Poisson ensemble $\mathcal{P}_k(\alpha, n)$. The dimension of the code is $m = \alpha n$ and the rate is $r = 1/\alpha$. The code can also be seen as a planted k-XOR-SAT formula.

Despite the long history of research on the LDPC and LDGM codes, and their success in practical applications of communications, there are still many open

questions concerning the behaviour of these codes. In particular, even for the simple code described above, it is still open to show that the mutual information $\frac{1}{n}I(X^n;Y^m)$ concentrates, with the exception of the binary erasure channel for which much more is known [35,34]. In the case of dense random codes, standard probability arguments show that concentration occurs with a transition at capacity for any discrete memoryless channels. But for sparse codes, the traditional arguments fail. Recently, the following was conjectured in [43] for constant right degree LDGM codes and binary input symmetric output channels [5],

$$\mathbb{P}\{\frac{1}{m}I(X;Y) < C(W)\} \to \begin{cases} 0 & \text{if } \alpha < C_k(W) \\ 1 & \text{if } \alpha > C_k(W) \end{cases} \tag{18}$$

where $C_k(W)$ is a constant depending on k and W.

We provide next a concentration result for this model, which implies the above conjecture for even degrees.

Definition 5. *An encoded symmetric kernel is given by*

$$Q(z|u) = W(z|\oplus_{i=1}^{k} u_i), \tag{19}$$

where W is a binary input symmetric output (BISO) channel from $\mathcal{X}$ to $\mathcal{Y}$.

Note that this corresponds to the output of a BISO W when the input to the channel is the XOR of k information bits. This corresponds also to the constant right-degree LDGM codes considered in the conjecture of [43].

Lemma 7. *For an encoded symmetric kernel with k even, the operator Γ_l is convex for any $l \geq 1$.*

Corollary 5. *Let X be uniformly drawn in $GF(2)^n$, $U = XG$ be the output of a k-degree LDGM code of dimension αn, and Y be the output of U on a BISO channel W. Then $\frac{1}{n}I(X;Y)$ converges in probability to a constant $C_k(\alpha, W)$.*

Note that

$$\frac{1}{m}I(X;Y) = \frac{1}{m}H(Y) - H(W), \tag{20}$$

where $H(W)$ denotes the conditional entropy of the channel W. Hence

$$\frac{1}{m}I(X;Y) < 1 - H(W) \equiv \frac{1}{m}H(Y) < 1. \tag{21}$$

Since $\frac{1}{m}H(Y)$ converges from previous corollary, and since the limit must be decreasing in α (increasing in r), the conjecture (18) follows.

[5] This means that the channel is a weighted sum of binary symmetric channels.

5 Proof Outline for Theorem 1: Interpolation Method for Graphical Channels

We now show the sub-additivity of $H^{(n)}(X|Y)$,

$$H^{(n)}(X|Y) \leq H^{(n_1)}(X|Y) + H^{(n_2)}(X|Y). \tag{22}$$

Note that if we partition the set of vertices $[n]$ into two disjoint sets of size n_1 and n_2 with $n_1 + n_2 = n$, and denote by g_1 and g_2 the restriction of g onto these subsets obtained by **removing** all the crossing hyperedges, then the following is obtained by basic properties of the entropy

$$H_g^{(n)}(X|Y) \leq H_{g_1}^{(n)}(X|Y) + H_{g_2}^{(n)}(X|Y). \tag{23}$$

Hence the above is true for a random graph G drawn from the ensemble $\mathcal{P}_k(\alpha, n)$. However, the terms $H_{G_i}^{(n)}(X|Y)$, $i = 1, 2$, do not correspond to $H_{G_i}^{(n_i)}(X|Y)$, since the edge probability is $\frac{\alpha n}{\binom{n}{k}}$ and not $\frac{\alpha n_i}{\binom{n_i}{k}}$. Consequently, the above does not imply $H^{(n)}(X|Y) \leq H^{(n_1)}(X|Y) + H^{(n_2)}(X|Y)$. To obtain the proper term on the right hand side, one should add the edges lost in the splitting of the vertices (e.g., via a coupling argument), but this gives a lower bound on the right hand side of (23), conflicting with the upper bound. The interpolation method provides a way to compare the right quantities.

The interpolation method was first introduced in [26] for the Sherrington-Kirkpatrick model. This is a model for a spin-glass (i.e. a spin model with random couplings) on a complete graph. It was subsequently shown in [20,21,42] that the same ideas can be generalized to models on random sparse graphs, and applications in coding theory and random combinatorial optimization were proposed in [37,30] and [10,1]. We next develop an interpolation method to estimate the conditional entropy of general graphical channels for different values of n. Interestingly, the planting flips the behaviour of the entropy from supper to sub-additive.

Definition 6. *We define a more general Poisson model for the random graph, where a parameter $\varepsilon_I \geq 0$ is attached to each $I \in E_k(V)$, and the number of edges $m_I(\varepsilon_I)$ is drawn from a Poisson distribution of parameter ε_I. This defines a random hypergraph whose edge probability is not homogenous but depends on the parameters ε_I. Denoting by $\underline{\varepsilon}$ the collection of all $\binom{n}{k}$ parameters ε_I, we denote this ensemble as $\mathcal{P}_k(\underline{\varepsilon}, n)$. If for any I, $\varepsilon_I = \frac{\alpha n}{\binom{n}{k}}$, $\mathcal{P}_k(\underline{\varepsilon}, n)$ reduces to $\mathcal{P}_k(\alpha, n)$ as previously defined.*

Lemma 8. *Let X be uniformly drawn over $\mathcal{X}^n$, G be a random hypergraph drawn from the ensemble $\mathcal{P}_k(\underline{\varepsilon}, n)$ independently of X, and $Y(\underline{\varepsilon})$ be the output of X through P_G defined in (1) for a kernel Q. Then*

$$\frac{\partial}{\partial \varepsilon_I} H^{(n)}(X|Y(\underline{\varepsilon})) = -I(Y_I; X_I|Y(\underline{\varepsilon}))\,, \tag{24}$$

where Y_I and $Y(\underline{\varepsilon})$ are independent conditionally on X (i.e., Y_I is drawn under $Q(\cdot|X[I])$ and $Y(\underline{\varepsilon})$ is drawn independently under $R_G(\cdot|X)$).

We define a *path* as a differentiable map $t \mapsto \underline{\varepsilon}(t)$, with $t \in [0,T]$ for some $T \geq 0$. We say that a path is balanced if

$$\sum_{I \in E_k(V)} \frac{\mathrm{d}\varepsilon_I}{\mathrm{d}t}(t) = 0\,. \tag{25}$$

We will write $\dot{\varepsilon}_I(t)$ for the derivative of $\varepsilon_I(t)$ along the path and $Y(t)$ for $Y(\underline{\varepsilon}(t))$.

Corollary 6. *For a balanced path*

$$\frac{\mathrm{d}}{\mathrm{d}t} H(X|Y(t)) = -\sum_{I \in E_k(V)} H(Y_I|Y(t))\, \dot{\varepsilon}_I(t)\,. \tag{26}$$

Given a partition $V = V_1 \sqcup V_2$, we define the associated *canonical path* $\underline{\varepsilon} : t \in [0,1] \to \underline{\varepsilon}(t) \in [0,1]^{E_k(V)}$ as follows. Let $n_i = |V_i|$, $m_i = |E_k(V_i)|$, $i \in \{1,2\}$, and $m = |E_k(V)|$. We define

$$\varepsilon_I(0) \equiv \frac{\alpha n}{m}, \quad \forall I \in E_k(V), \tag{27}$$

$$\varepsilon_I(1) \equiv \begin{cases} \frac{\alpha n_1}{m_1} & \text{if } I \in E_k(V_1) \\ \frac{\alpha n_2}{m_2} & \text{if } I \in E_k(V_2) \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

and

$$\underline{\varepsilon}(t) = (1-t)\underline{\varepsilon}(0) + t\underline{\varepsilon}(1). \tag{29}$$

Note that the canonical path is balanced. Moreover, at time $t = 0$, $\mathcal{P}_k(\underline{\varepsilon}(0), n)$ reduces to the original ensemble $\mathcal{P}_k(\alpha, n)$, and at time $t = 1$, $\mathcal{P}_k(\underline{\varepsilon}(1), n)$ reduces to two independent copies of the original ensemble on the subset of n_1 and n_2 variables: $\mathcal{P}_k(\alpha, n_1) \times \mathcal{P}_k(\alpha, n_2)$.

Applying Lemma 6, we obtain the following.

Corollary 7. *For the canonical path*

$$\frac{\mathrm{d}}{\mathrm{d}t} H(X|Y(t)) = \alpha n \mathbb{E}_I H(Y_I|Y(t)) - \alpha n_1 \mathbb{E}_{I_1} H(Y_{I_1}|Y(t)) - \alpha n_2 \mathbb{E}_{I_2} H(Y_{I_2}|Y(t))\,, \tag{30}$$

where I is drawn uniformly in $E_k(V)$, and I_i, $i \in \{1,2\}$, are drawn uniformly in $E_k(V_i)$.

We recall that

$$H(Y_I|Y(t)) = -\mathbb{E}_{Y,Y_I} \log \sum_x Q(Y_I|x[I]) R_{G(t)}(x|Y) \tag{31}$$

$$= -\mathbb{E}_{Y(t),Y_I} \log \mathbb{E}_{X|Y(t)} Q(Y_I|X[I])\,, \tag{32}$$

where $Y(t)$ is the output of $P_{G(t)}$ and $\mathbb{E}_{X|Y(t)}$ is the conditional expectation over $R_{G(t)}$.

Lemma 9

$$\frac{1}{\alpha|\mathcal{Y}|}\frac{\mathrm{d}}{\mathrm{d}t}H(X|Y(t)) = -\sum_{l=2}^{\infty}\frac{1}{l(l-1)}\mathbb{E}_{X^{(1)},\dots,X^{(l)}}\left[n\Gamma_l(V) - n_1\Gamma_l(V_1) - n_2\Gamma_l(V_2)\right] \tag{33}$$

where

$$\Gamma_l(V) \equiv \mathbb{E}_{I,W_I}\prod_{r=1}^{l}\left(1 - P(W_I|X^{(r)}[I])\right), \tag{34}$$

I is uniformly drawn in $E_k(V)$, W_I is uniformly drawn in $\mathcal{Y}$, and $X^{(1)},\dots,X^{(l)}$ are drawn under the probability distribution

$$\sum_{y}\prod_{i=1}^{l}R_{G(t)}(x^{(i)}|y)\sum_{u}P_{G(t)}(y|u)2^{-n}. \tag{35}$$

This means that $X^{(1)},\dots,X^{(l)}$ are drawn i.i.d. from the channel $R_{G(t)}$ given a hidden output Y, these are the 'replica' variables, which are exchangeable but not i.i.d.. Note that denoting by ν the empirical distribution of $X^{(1)},\dots,X^{(l)}$, the above definition of $\Gamma_l(V)$ coincides with that of $\Gamma_l(\nu)$, hence the abuse of notation with definition (8). Hypothesis H ensures that Γ_l is convex for any distribution on $\mathcal{X}^l$, hence in particular for the empirical distribution of the replicas. Therefore, previous lemma implies Lemma 1 and Theorem 1 follows by the sub-additivity property.

References

1. Abbe, E., Montanari, A.: On the concentration of the number of solutions of random satisfiability formulas. Random Structures & Algorithms (2013) ISSN: 1098-2418, `http://dx.doi.org/10.1002/rsa.20501`, doi:10.1002/rsa.20501
2. Abbe, E., Montanari, A.: Conditional Random Fields, Planted Constraint Satisfaction, and Entropy Concentration. `arXiv:1305.4274 [math.PR]` (2013)
3. Achlioptas, D., Coja-Oghlan, A.: Algorithmic barriers from phase transitions. In: Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, pp. 793–802. IEEE Computer Society, Washington, DC (2008)
4. Krivelevich, M., Coja-Oghlan, A., Vilenchik, D.: Why almost all satisfiable k-cnf formulas are easy. In: Proceedings of the 13th International Conference on Analysis of Algorithms, pp. 89–102 (2007)
5. Achlioptas, D., Jia, H., Moore, C.: Hiding satisfying assignments: two are better than one. In: Proceedings of AAAI 2004, pp. 131–136 (2004)
6. Achlioptas, D., Kautz, H., Gomes, C.: Generating satisfiable problem instances
7. Achlioptas, D., Han Kim, J., Krivelevich, M., Tetali, P.: Two-coloring random hypergraphs. Random Structures and Algorithms 20(2), 249–259 (2002)
8. Altarelli, F., Monasson, R., Zamponi, F.: Can rare SAT formulas be easily recognized? On the efficiency of message passing algorithms for K-SAT at large clause-to-variable ratios. Computing Research Repository abs/cs/060 (2006)

9. Achlioptas, D., Naor, A., Peres, Y.: Rigorous Location of Phase Transitions in Hard Optimization Problems. Nature 435, 759–764 (2005)
10. Bayati, M., Gamarnik, D., Tetali, P.: Combinatorial approach to the interpolation method and scaling limits in sparse random graphs. In: 42nd Annual ACM Symposium on Theory of Computing, Cambridge, MA, pp. 105–114 (June 2010)
11. Barthel, W., Hartmann, A.K., Leone, M., Ricci-Tersenghi, F., Weigt, M., Zecchina, R.: Hiding solutions in random satisfiability problems: A statistical mechanics approach. Phys. Rev. Lett. 88, 188701 (2002)
12. Berlekamp, E., McEliece, R.J., Van Tilborg, H.C.A.: On the inherent intractability of certain coding problems (corresp.). IEEE Transactions on Information Theory 24(3), 384–386 (1978)
13. Coja-oghlan, A.: Graph partitioning via adaptive spectral techniques. Comb. Probab. Comput. 19(2), 227–284 (2010)
14. Dyer, M.E., Frieze, A.M.: The solution of some random np-hard problems in polynomial expected time. Journal of Algorithms 10(4), 451–489 (1989)
15. Dietzfelbinger, M., Goerdt, A., Mitzenmacher, M., Montanari, A., Pagh, R., Rink, M.: Tight thresholds for cuckoo hashing via XORSAT. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part I. LNCS, vol. 6198, pp. 213–225. Springer, Heidelberg (2010)
16. Decelle, A., Krzakala, F., Moore, C., Zdeborová, L.: Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. Phys. Rev. E 84, 066106 (2011)
17. Dubois, O., Mandler, J.: The 3-XORSAT threshold. In: Proceedings of the 43rd Symposium on Foundations of Computer Science, FOCS 2002, pp. 769–778. IEEE Computer Society, Washington, DC (2002)
18. Daudé, H., Ravelomanana, V.: Random 2-XORSAT at the satisfiability threshold. In: Laber, E.S., Bornstein, C., Nogueira, L.T., Faria, L. (eds.) LATIN 2008. LNCS, vol. 4957, pp. 12–23. Springer, Heidelberg (2008)
19. Elias, P.: Coding for noisy channels. IRE Convention Record 4, 37–46 (1955)
20. Franz, S., Leone, M.: Replica bounds for optimization problems and diluted spin systems. J. Stat. Phys. 111, 535 (2003)
21. Franz, S., Leone, M., Toninelli, F.L.: Replica bounds for diluted non-Poissonian spin systems. J. Phys. A 36, 10967 (2003)
22. Feige, U., Mossel, E., Vilenchik, D.: Complete convergence of message passing algorithms for some satisfiability problems. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX and RANDOM 2006. LNCS, vol. 4110, pp. 339–350. Springer, Heidelberg (2006)
23. Fortunato, S.: Community detection in graphs. Physics Reports 486(3-5), 75–174 (2010)
24. Friedgut, E.: Sharp thresholds of graph properties, and the k-sat problem. J. Amer. Math. Soc. 12, 1017–1054 (1999); appendix by Bourgain, J.
25. Gallager, R.G.: Low-density parity-check codes. MIT Press, Cambridge (1963)
26. Guerra, F., Toninelli, F.L.: The thermodynamic limit in mean field spin glasses. Commun. Math. Phys. 230, 71–79 (2002)
27. Goldenberg, A., Zheng, A.X., Fienberg, S.E., Airoldi, E.M.: A survey of statistical network models. Foundations and Trends® in Machine Learning 2(2), 129–233 (2010)
28. Haanpää, H., Järvisalo, M., Kaski, P., Niemelä, I.: Hard satisfiable clause sets for benchmarking equivalence reasoning techniques (2005)
29. Jia, H., Moore, C., Strain, D.: Generating hard satisfiable formulas by hiding solutions deceptively. In: AAAI, pp. 384–389. AAAI Press (2005)

30. Kudekar, S., Macris, N.: Sharp bounds for optimal decoding of Low-Density Parity-Check codes. IEEE Trans. on Inform. Theory 55, 4635–4650 (2009)
31. Karrer, B., Newman, M.E.J.: Stochastic blockmodels and community structure in networks. Phys. Rev. E 83, 016107 (2011)
32. Krzakala, F., Zdeborová, L.: Hiding quiet solutions in random constraint satisfaction problems. Phys. Rev. Lett. 102, 238701 (2009)
33. Lafferty, J.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data, pp. 282–289. Morgan Kaufmann (2001)
34. Luby, M., Mitzenmacher, M., Shokrollahi, A., Spielman, D.A., Stemann, V.: Practical loss-resilient codes. In: 29th Annual ACM Symposium on Theory of Computing, pp. 150–159 (1997)
35. Luby, M., Mitzenmacher, M., Shokrollahi, A., Spielman, D.A.: Efficient erasure correcting codes. IEEE Trans. on Inform. Theory 47(2), 569–584 (2001)
36. Mossel, E., Neeman, J., Sly, A.: Stochastic Block Models and Reconstruction. arXiv:1202.1499 [math.PR]
37. Montanari, A.: Tight bounds for LDPC and LDGM codes under MAP decoding. IEEE Trans. on Inform. Theory 51, 3221–3246 (2005)
38. Montanari, A.: Estimating random variables from random sparse observations. European Transactions on Telecommunications 19(4), 385–403 (2008)
39. Montanari, A., Restrepo, R., Tetali, P.: Reconstruction and Clustering in Random Constraint Satisfaction Problems. CoRR abs/0904.2751 (2009)
40. Newman, M.E.J.: Communities, modules and large-scale structure in networks. Nature Physics 8(1), 25–31 (2011)
41. Pittel, B., Sorkin, G.B.: The Satisfiability Threshold for k-XORSAT. arXiv:1212.1905 (2012)
42. Panchenko, D., Talagrand, M.: Bounds for diluted mean-field spin glass models. Prob. Theor. Rel. Fields 130, 319–336 (2004)
43. Raj Kumar, K., Pakzad, P., Salavati, A.H., Shokrollahi, A.: Phase transitions for mutual information. In: 2010 6th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), pp. 137–141 (2010)
44. Richardson, T., Urbanke, R.: Modern Coding Theory. Cambridge University Press, Cambridge (2008)
45. Zdeborová, L., Krzakala, F.: Quiet planting in the locked constraint satisfaction problems. SIAM Journal on Discrete Mathematics 25(2), 750–770 (2011)

Finding Heavy Hitters from Lossy or Noisy Data

Lucia Batman, Russell Impagliazzo*, Cody Murray**,
and Ramamohan Paturi**

Department of Computer Science and Engineering,
University of California, San Diego

Abstract. Motivated by Dvir et al. and Wigderson and Yehudayoff [3,10], we examine the question of discovering the set of heavy hitters of a distribution on strings (i.e., the set of strings with a certain minimum probability) from lossy or noisy samples. While the previous work concentrated on finding both the set of most probable elements and their probabilities, we consider *enumeration*, the problem of just finding a list that includes all the most probable elements without associated probabilities. Unlike Wigderson and Yehudayoff [10], we do not assume the underlying distribution has small support size, and our time bounds are independent of the support size. For the enumeration problem, we give a polynomial time algorithm for the lossy sample model for any constant erasure probability $\mu < 1$, and a quasi-polynomial algorithm for the noisy sample model for any *noise* probability $\nu < 1/2$ of flipping bits. We extend the lower bound for the number of samples required for the reconstruction problem from [3] to the enumeration problem to show that when $\mu = 1 - o(1)$, no polynomial time algorithm exists.

Keywords: Population Recovery, Enumeration of Heavy Hitters, Learning Discrete Distributions.

1 Introduction

Say that you are an investigator investigating DNA evidence at a crime scene. You can collect and analyze random DNA strands available at the scene, and you want to find DNA from each person involved in the crime, whether perpetrator or victim. There are several complications that make your task more difficult. First, it is not possible to exhaustively search through the huge number of microscopic strands of DNA at the scene; the best you can do is randomly pick strands and sequence them. Secondly, much of the DNA might have nothing to do with the crime. There might be small amounts of trace DNA from a huge number of people who passed by the scene at some time before the crime took place,

* Work supported by the Simons Foundation, the Ellentuck Fund, the Friends of the Institute for Advanced Study, and NSF grants DMS-0835373, CCF-121351 and CCF-0832797 subcontract No. 00001583.

** This research is supported by NSF grant CCF-1213151 from the Division of Computing and Communication Foundations.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 347–362, 2013.

and there might be contamination later. Thirdly, even the DNA from the people involved will be only partially recoverable, with either missing pieces or random noise.

Ideally, a complete crime scene analysis would give not just which DNA sequences were found, but their relative proportions. However, it would also be extremely useful to know the set of sequences present. You couldn't hope to get a complete list of trace sequences without sequencing every single strand. But you would want to filter out trace elements anyway, to concentrate on those from likely suspects.

This example illustrates the general issue of trying to analyze a distribution from lossy or noisy samples. Such problems are central to statistics, and arise in a variety of scientific circumstances. In theoretical computer science, Kearns et al. [6] introduced the general question of when a distribution can be identified from samples, and gave the first algorithms for formulations of the problem above. After that, attention was mainly focused on the continuous version of learning mixtures of Gaussians, which had been introduced by the statistician Pearson in the nineteenth century and was already a subject of great interest in AI and statistics. This problem spurred some highly interesting and deep algorithmic work (for example, [2,9,8,1]).

The equally fascinating case of lossy or noisy discrete distributions, e.g., lossy or noisy distributions on strings, only started getting attention again relatively recently. In particular, Dvir et al. [3] used an algorithm to infer a distribution on strings from lossy samples as a sub procedure in a learning algorithm for DNFs in the restriction model. Their goal was to learn the underlying distribution by giving an explicit description of a distribution which is close to the distribution that the samples were drawn from. For this reason, the problem as they formalized it assumed that the distribution had small support, with at most k non-zero probability elements, for a known parameter k which could affect the running time of the algorithm. However, none of the techniques they used relied on this assumption. On the other hand, the quasi-polynomial recovery algorithm of Wigderson and Yehudayoff [10] from noisy samples does rely on the fact that the support size is small.

In this work, we introduce a goal that is less ambitious but potentially more robust. There are many situations, like the crime scene investigation mentioned above, when it is desirable to identify a set containing all the probable elements, but it is not necessary to provide a complete description of the distribution. For example, perhaps one wants to identify gene patterns that are relatively common among the DNA of drug-resistant bacteria, texts that are most duplicated on web pages, or snippets of code that appear in many computer viruses. This problem of identifying the "heavy hitters" of a distribution, has been extensively studied in the context of streaming algorithms, see, for example, [5]. While just identifying the heavy hitters requires less information than exactly characterizing the distribution, for the same reason, it makes sense even when the distribution has no nice description, e.g., when the distribution has a constant fraction of mass divided among a very large, arbitrary set of strings. In this sense, algorithms for

heavy hitters can be *agnostic* in the sense of agnostic learning. We wish to find the best fit to a distribution with small support without relying on the assumption that the distribution actually has small support.

We consider two basic error models. In the lossy sample model, each bit of the sample drawn from the distribution is independently erased with probability μ. For the lossy model, we provide a polynomial time algorithm that identifies heavy hitters for any constant erasure probability $\mu < 1$. The error models for which our algorithm works are very general, and we do not require independence of noise in a bit from either the noise in other bits or the input, as long as the chance of erasure is never too large. For the case of independent erasures, the estimation algorithms of [3,7] can also solve the enumeration problem. As a converse, we also show that, if the erasure probability is $1 - o(1)$, no polynomial time algorithm exists for enumerating the heavy hitters.

In the noisy sample model, each bit of the sample drawn from the distribution is independently flipped with probability ν. For the noisy sample model, we give a quasi-polynomial time algorithm for identifying heavy hitters, for any constant $\nu < 1/2$. This algorithm is related but incomparable to the quasi-polynomial distribution recovery algorithm of Dvir et al. [10]. On the one hand, our algorithm is considerably faster (our exponent is doubly logarithmic rather than logarithmic), and does not require the distribution to have small support. On the other hand, they solve the harder problem of estimating the probabilities for each heavy hitter.

1.1 Problem Definitions and Formal Statements of Our Results

The general issue of learning about distributions from noisy samples has many variations and interesting formulations. Below, we will consider the ingredients of such formulations and how they relate to each other.

Underlying distribution. In all versions, the samples come from some underlying distribution D on $\{0,1\}^n$. D is unknown to the algorithm. The algorithm can only access samples drawn from the D after errors are introduced. As mentioned earlier, [3,10] make the assumption that D has support size at most k, where k is given to the algorithm. Run times are given in terms of k and n. We call this the *small support size* case. We mainly consider the case when D is *arbitrary*. One could also consider other families of distributions, e.g., easily sampled distributions, high-entropy distributions or distributions that consist of independent pairs of strings x, y.

Error models. Like [3,10], we examine two basic types of error. The basic *lossy error* model is as follows. After a string $x = x_1 \dots x_n$ is drawn from D, the observed sample has the form $y_1 \dots y_n$ where for each i independently, we set $y_i = *$ with probability μ, and $y_i = x_i$ otherwise, where $*$ is a new symbol. The difficulty of the problem in the lossy model depends on the constant μ, the larger the μ the more difficult the problem is. Some of our algorithms and one of the algorithms in Dvir et al. [3] only work for μ smaller than a specified constant.

In the *noisy* model, the observed sample has the form $y_1 \dots y_n$ where for each i independently, we set $y_i = 1 - x_i$ with probability ν, and $y_i = x_i$ otherwise.

We can also consider the problem for other error models. For example, our lossy enumeration algorithm works equally well in a semi-random erasure model, where, after erasures occur, an adversary is allowed to "un-erase" an arbitrary set of positions. It is easy to see that estimation is not possible in such a model.

Goal. We distinguish three goals for an algorithm. We are primarily concerned with the *enumeration* problem. Here, the algorithm has as input a parameter $\epsilon > 0$, and needs to output a list L that contains all ϵ-heavy hitters of D, i.e., all the strings z so that $\mathbf{P}[x = z] \geq \epsilon$. L may contain some non-heavy hitters, but we expect the algorithm to explicitly list the elements of L, so an efficient algorithm cannot output a large list. There may in general be $1/\epsilon$ heavy hitters, so an ideal algorithm runs in polynomial time in n and $1/\epsilon$.

In the *estimation* problem, the algorithm is given z and ϵ, and is required to output an estimate of $\mathbf{P}[x = z]$ that is correct within additive error ϵ. Even without errors, getting such an estimate would require $1/\epsilon^2$ samples, so again polynomial time in n and $1/\epsilon$ is the best we could hope for.

Finally, in the *recovery* problem, we wish to find a list of elements that contains all 2ϵ-heavy hitters and only contains ϵ-heavy hitters, and for each element on our list we wish to have an estimate of its probability that is within an additive error of at most ϵ. Note that, for distributions with support k, a recovery algorithm with $\epsilon = \delta/k$ will provide an explicit description of a distribution that is within δ of the actual one. This is the sense that a recovery algorithm actually "recovers" the original distribution.

The recovery problem combines the enumeration and estimation problems, but Dvir et al. [3] observe that any estimation algorithm can be used to solve the recovery problem with only a factor n overhead, via a "branch-and-prune" approach. Thus, we only need to actually look at enumeration and estimation, and estimation is the more difficult problem.

We devise or analyze algorithms for several of these problems. The results we present in this extended abstract are:

1. For any constant $\mu < 1$, we give a polynomial time (in n and $1/\epsilon$) algorithm for enumeration of the ϵ-heavy hitters from lossy samples where μ is the bit erasure probability. Here, the distribution is arbitrary and the time does not depend on the support size of the distribution. The list size is a polynomial in $1/\epsilon$ that does not depend on n.

   In fact, we need only the following property of the erasure model: There is a parameter T so that, for any subset of T bit positions, the probability that all bits are erased is $o(\epsilon^2)$, whereas the probability that no bit is erased is at least $poly(\epsilon)$. For independent erasures, both are true when $T = C \log 1/\epsilon$, where C is a constant depending on μ. But this property will also be inherited by any samples with fewer erasures, such as the semi-random distribution

mentioned above. It will also hold for e.g., T-wise independent distributions on erasures.
2. For samples with independent noise ν for any constant $\nu < 1/2$, we give an enumeration algorithm that takes polynomial time in n, and quasi-polynomial time in $1/\epsilon$ (more precisely, time $\text{poly}(n)\epsilon^{O(\log\log\epsilon)}$) time . Here, the distribution is arbitrary and the time does not depend on the support size of the distribution. The list size does not depend on n.
3. [3] give a super-polynomial lower bound for estimation from lossy samples even for small support distributions when $\mu = 1 - o(1)$. We give a similar lower bound for enumeration from lossy samples for small support distributions.
4. Dvir et al. [3] also observe that LP-duality can be used to characterize the sample complexity of algorithms for estimation for arbitrary distributions, for any error model. We give a relaxation of this LP program that shows a "Yao principle" for such estimation problems. We show that either there is an algorithm for estimation using a certain number of samples or a polynomially-related lower bound via two distributions whose noisy versions are indistinguishable. [3] also introduce the notion of "local inverse" to a matrix. We show that the existence of any algorithm for estimation for arbitrary distributions in an error model implies one via a local inverse to the corresponding matrix.

In the full paper, we will include some additional results, not directly related to enumeration:

1. We give a new algorithm for estimation from lossy samples that works in time polynomial in n and exponential in $1/\epsilon$ whenever $\mu \leq 2/3$.
2. Dvir et al. [3] have shown that their estimation algorithm for lossy samples will work for arbitrary distributions for $\mu < 0.614....$ We give a tighter analysis of their algorithm, showing that it works when $\mu < 1 - 1/\sqrt{2} = 0.69....$ We present numerical computations that suggest that the real threshold for this algorithm is $\mu = .75$.

1.2 State-of-the-Art

For lossy samples with independent erasures, the current best algorithms for all versions of the problem are due to Moitra and Saks [7], who, subsequently to our work, solve the hardest version of this problem. They give a polynomial time algorithm for estimation for arbitrary distributions, which implies similar algorithms for enumeration and recovery. This result improves on the estimation and recovery algorithms of [3], and is better than the algorithms from the first two of our additional results to appear in the full paper. Our enumeration algorithm is still useful if the erasures are not actually independent.

For noisy samples, for distributions of small support size, the current best algorithm for estimation and recovery problems is the quasi-polynomial algorithm of [10]. For any constant level of noise, this algorithm runs in time polynomial in n and $k^{\log k}$ when the support size is k. Our second result above is the only

non-trivial algorithm known algorithm for enumeration when the distribution is arbitrary, and is the fastest algorithm for enumeration even when the support is small. For the estimation and recovery problems when the distributions are arbitrary, no algorithms better than exponential time in n are known.

Combining the lower bound of [3] and the lower bound in result three above, all versions of these problems require more than polynomially many samples when the bit erasure probability $\mu = 1 - o(1)$ or when the bit flipping probability is $\nu = 1/2 - o(1)$.

2 Branch-and-Prune Algorithms

Our algorithms for enumeration follow the same "branch-and-prune" paradigm as those of [3,10]. This paradigm can be viewed as a form of classical dynamic programming [11], but in the context of probabilistic algorithms for enumeration, the first example we know of is the Goldreich-Levin algorithm for list decoding the Hadamard code. [4]. Dvir et al. [3] used a branch-and-prune method to reduce the recovery problem to the estimation problem. In this section, we revisit this connection and present a self-contained explanation. We also emphasize the minimal set of conditions that a pruning algorithm needs to have to be useful for enumeration.

Let D be the underlying distribution on n bit strings. For $1 \leq m \leq n$, let D_m be the distribution on the first m bits of a string drawn from D. In the context of enumerating heavy hitters, we observe that the distribution on the first m bits of the observed sample is equal to the distribution on lossy/noisy samples from D_m with the same error parameter μ or ν. Also note that the m-bit prefix of any heavy hitter for D is a heavy hitter for D_m. Our goal is, for m from 1 to n, to find a set S_m of candidates that contains all the heavy hitters of D_m. Then the heavy hitters for D_{m+1} are contained in the set $T_{m+1} = \{x0 | x \in S_m\} \cup \{x1 | x \in S_m\}$. This is the "branch" stage. However, to prevent this set of candidates from growing exponentially, it is necessary to "prune" this set back to a smaller subset S_{m+1}.

To be more precise, a *pruning algorithm* takes a set of m-bit strings T and can request lossy or noisy samples from D_m, and produces a subset $S \subset T$. We say the strings in $T - S$ are *pruned* by the algorithm. Our pruning algorithms have a parameter s and we need two properties to ensure their efficiency:

Correctness. With high probability $(1 - o(1/(ns)))$, no heavy hitter for D_m is pruned.

Efficiency. If $|T| > s$, then with high probability S is a strict subset of T.

Note that an estimation algorithm can be used as a pruning algorithm: Estimate the probability of each element of T to within an additive term of $\epsilon/3$. Prune the ones whose estimates are less than $2/3\epsilon$. Assuming the estimates are correct, all ϵ heavy hitters are maintained. Furthermore, all but at most $s = 3/\epsilon$ elements with probability at least $\epsilon/3$ are pruned.

If both of these conditions hold, we can use the pruning algorithm for enumeration. We maintain a set S_m that with high probability contains all heavy hitters

for D_m. We construct T_{m+1} as above, and prune it until the size remaining is at most s. Since T_{m+1} has at most $2s$ elements at the start, and each time we prune we get a strict subset, pruning can happen at most s times for any m. The probability of pruning a heavy hitter in any one run of the pruning algorithm is $o(1/(sn))$ so with high probability, we never prune a heavy hitter. We will need to use the pruning algorithm $O(ns)$ times on a set of strings of size $O(s)$.

If we have an estimation algorithm and an enumeration algorithm, we can enumerate a list S_n of heavy hitters and estimate the probability of each of the candidate heavy hitters in S_n to get a recovery algorithm.

Thus, it suffices to describe an algorithm for estimation or pruning in order to specify an algorithm for recovery or enumeration, respectively.

3 Enumeration Algorithms

In this section, we give a polynomial time enumeration algorithm for lossy samples for any $\mu < 1$, and a quasi-polynomial time algorithm for noisy samples for any $\nu < 1/2$. As described before, we only need to give a pruning algorithm in each case.

3.1 Lossy Samples

We describe here the pruning algorithm for lossy samples. Let $\epsilon > 0$. For some polynomial $s = n^{O(1)}\epsilon^{-O(1)}$, we are given a set T of at most $2s$ strings that includes all ϵ-heavy hitters, and want to find a subset S that contains all ϵ-heavy hitters, but has at most s strings. When the parameter ϵ is clear from the context, we will simply refer to an ϵ-heavy hitter as a heavy hitter.

The pruning algorithm works in two phases. In the Phase I, we will collect a maximal set $B \subseteq T$ of *centers* of small size such that the centers are at a distance of at least $d = C \log \frac{4}{\epsilon}$ from each other, where C is a constant which only depends on μ and will be determined later. Since B is maximal, each heavy hitter is either in B or at most at a distance of d from an element in B. In the second phase, for each center, we consider the elements of T close to it and prune them.

Phase I: In Phase I, we start with a set B of centers (initially an empty set) of size no more than $2/\epsilon$ and greedily place non-pruned strings from T into B as long as $|B| \leq 4/\epsilon$ so that all centers in B are at least a distance d from each other. If the size of B never equals $\lceil 4/\epsilon \rceil$, we proceed to Phase II. If not, we execute the following sub procedure to prune the non-heavy hitters from B so that the size to B reduces to no more than $2/\epsilon$. We then repeat the greedy of process of growing B.

Pruning B: At this point, we have $|B| = 4/\epsilon$. Our goal is to cut the size of B by half by pruning away sufficiently many non-heavy centers of B.

For $v, w \in T$ define $\Delta_{v,w} = \{i | v_i \neq w_i\}$. In other words, $\Delta_{v,w}$ is the set of positions where the strings v and w differ. For $u, u' \in B$ and for a lossy sample

y, we say that y is (u, u')-*discriminatory* if for at least one position in $\Delta_{u,u'}$, the value is not erased in y. We say y is *discriminatory* if it is (u, u')-discriminatory for all $u, u' \in B$. y is *consistent* with a string $u \in T$ if it never disagrees with u in a revealed bit. Note that y is always consistent with the original sample.

To prune non-heavy strings in B, we draw $t = \text{poly}(1/\epsilon)$ lossy samples. For each center u, we compute the fraction p_u of samples that are discriminatory and consistent with u. Each sample can contribute to at most one center since a discriminatory sample can be consistent with at most one center. If a discriminatory sample y were to be consistent with centers u and u', then there is a position in $\Delta_{u,u'}$ which is not erased in y where u and u' agree, a contradiction. Finally, if $p_u < \epsilon/2$, we prune u.

We will argue that at least $2/\epsilon$ strings of B are pruned by the procedure so we will end up with a B of size at most $2/\epsilon$. Let y be a lossy sample. y is discriminatory with probability at least $7\epsilon/8$ since

$$\begin{aligned}\mathbf{P}[y \text{ is not discriminatory}] &\leq \sum_{u,u' \in B} \mathbf{P}[y \text{ is not } (u,u')\text{-discriminatory}] \\ &\leq (8/\epsilon^2)\mu^d \leq (8/\epsilon^2)\mu^{C \log 4/\epsilon} \\ &= (8/\epsilon^2)(\epsilon^3/64) \text{ for } C = 3/\log \tfrac{1}{\mu} \\ &\leq \epsilon/8\end{aligned}$$

Let u be a ϵ-heavy hitter. The probability that y is discriminatory and is consistent with u is at least $7\epsilon/8$. Let p_u be the fraction of discriminatory samples consistent with u. $\mathbf{P}[p_u \leq \epsilon/2] \leq e^{-2(3\epsilon/8)^2 t}$ by Chernoff-Hoeffding bound since the expected fraction is at least $7\epsilon/8$. $\mathbf{P}[\exists$ an ϵ-heavy center u such that $p_u \leq \epsilon/2] \leq \frac{1}{\epsilon} e^{-2(3\epsilon/8)^2 t}$ by union bound since there are at most $1/\epsilon$ such centers. Since $t = \text{poly}(1/\epsilon)$, and since there are at most $2/\epsilon$ centers that pass the threshold of $\epsilon/2$, with high probability, we will have pruned at least $2/\epsilon$ centers while retaining all heavy hitters in B.

Phase II: At this point, we have a set B of centers of size at most $4/\epsilon$ such that every non-pruned element of T is within a distance of d from some center. Assign each non-pruned element of T to its closest center. For $u \in B$, let B_u denote the set of elements of T assigned to u.

For each u, we prune B_u so that its size is bounded by a polynomial in $1/\epsilon$. Since there are at most $4/\epsilon$ centers, we will end up with the desired bound after pruning. Fix $u \in B_u$. In the following, we outline how to prune B_u.

For $v \in B_u$, let Δ_v denote the set of positions where u and v differ. We say that a sample y is *revelatory* for v if for no position i in Δ_v, $y_i = *$.

We draw t lossy samples. For each v, we compute the fraction of samples that are revelatory for and consistent with v. Each sample can contribute to at most one element of B_u since a sample y can be revelatory for and consistent with at most one $v \in B_u$. For every $v, v' \in B_u$ and $v \neq v'$, there exists a position in $\Delta_v \cup \Delta_{v'}$ where one of v and v' agrees with u and the other disagrees with u. If y were to be revelatory for and consistent with both v and v', then v and v'

agree on every position in $\Delta_v \cup \Delta_{v'}$, which is a contradiction. If the fraction for v does not exceed the target $\epsilon^{C'+1}/2$, we prune it, where C' is a constant that depends only on μ.

Let $v \in B_u$ and y be a sample. We will show that the probability a sample y is revelatory for v is not small.

$$\begin{aligned}\mathbf{P}[y \text{ is revelatory for } v] &\geq (1-\mu)^{|\Delta_v|} \geq (1-\mu)^d = (1-\mu)^{3\log\frac{4}{\epsilon}/\log\frac{1}{\mu}} \\ &= (\epsilon/4)^{C'} \text{ for some constant } C' \text{ which only depends on } \mu\end{aligned}$$

Let $v \in B_u$ be an ϵ-heavy hitter. The probability that y is revelatory for and consistent with v is at least $(\epsilon/4)^{C'+1}$. Let p_v be the fraction of samples that are revelatory for and consistent with v. $\mathbf{P}[p_v \leq (\epsilon/4)^{C'+1}/2] \leq e^{-2((\epsilon/4)^{C'+1}/2)^2 t}$ by Chernoff-Hoeffding bound since the expected fraction is at least $(\epsilon/4)^{C'+1}$. $\mathbf{P}[\exists$ an ϵ-heavy v such that $p_v \leq (\epsilon/4)^{C'+1}/2] \leq \frac{1}{\epsilon} e^{-2((\epsilon/4)^{C'+1}/2)^2 t}$ by union bound since there are at most $1/\epsilon$ such elements. Since $t = \text{poly}(1/\epsilon)$, with high probability, at most $2/(\epsilon/4)^{C'+1}$ elements remain after pruning since there are at most $2/(\epsilon/4)^{C'+1}$ elements $v \in B_u$ such that $p_v \geq (\epsilon/4)^{C'+1}/2$.

A General Lossy Error Model: Our algorithm for enumerating ϵ-heavy hitters in the lossy error model and its analysis does not require that each bit position is erased independently with probability μ. Our algorithm and its analysis can be easily extended to lossy error models that satisfy weaker conditions.

Let x be an arbitrary sample drawn according to the original distribution D. Let y be a lossy model obtained from x according to a lossy error model. In the following we provide a sufficient condition for lossy error models that guarantees the correctness and performance of the above algorithm.

For all $\epsilon > 0$, there exists a t such that for all sets S of t positions,

1. $\mathbf{P}[$values in all positions of S are erased$|x] \leq \epsilon^3/64$, and
2. $\mathbf{P}[$none of the values in positions of S are erased$|x] \geq \epsilon^{O(1)}$.

3.2 Noisy Samples

Here we give a quasi-polynomial enumeration algorithm for the noisy case for any constant $0 \leq \nu < 1/2$. As before, we present just the pruning algorithm. As in the lossy case, it works in two phases. First, we locate a small number of *centers* so that every heavy hitter is $C \log 1/\epsilon$ distance from one of the centers, where C is a constant that only depends on μ. This already gives a $s = n^{O(\log 1/\epsilon))}$ algorithm, which works under a very general noise condition. However, we can improve it to $s = (1/\epsilon)^{O(\log(\log(1/\epsilon)))}$ in the second stage that requires noise to be exact and independent. In the second stage, for each center, we enumerate the heavy hitters within $d = O(\log 1/\epsilon)$ Hamming distance to the center. For each fixed center, this is equivalent to enumerating all of the low Hamming weight heavy hitters, which we can identify with the set of positions with value 1. To do this, we give a potential function for a set of positions A, which upper bounds the total probability of small Hamming weight strings that could contain A.

In addition to A, we'll maintain a set of positions R so that we only need to look at heavy hitters whose 1's are in R. We show that we can divide the extensions $A \cup \{i\}$ into two categories, one that significantly lowers the potential function, and another that significantly reduces the set R of positions we need to consider in the future. If the potential function drops below ϵ, we can prune the search. If R becomes very small, we can use a brute force search on small subsets of R.

The first phase works very similarly to the erasure case. We start with a set B of centers (initially an empty set) of size no more than $2/\epsilon$ and greedily place non-pruned strings from T into B as long as $|B| \leq 4/\epsilon$ so that all centers in B are at least a distance d from each other. If the size of B never equals $\lceil 4/\epsilon \rceil$, we proceed to phase two. If not, we execute the following sub procedure to prune the non-heavy hitters from B so that the size to B reduces to no more than $2/\epsilon$. We then repeat the greedy of process of growing B.

For $u, v \in B$, let $\Delta_{i,j}$ be the set of positions where u and v differ. We say that a noisy sample y *favors* $u \in B$ if for every $v \in B$, y agrees with u in strictly more than half the positions in $\Delta_{u,v}$. Note that, if u is itself is the original sample, each bit of the noisy sample y agrees with that of u with probability $1 - \nu > 1/2$. So the probability that at least half the positions of y in $\Delta_{u,v}$ disagree with u, by Chernoff-Hoeffding bound, is at most $e^{-\Omega(d(1/2-\nu)^2)} < \epsilon/16$ for some $C = O(1/(1/2 - \nu)^2)$. So by a union bound over the $4/\epsilon$ centers, if the original sample is u, the conditional probability that y does not favor u is at most $1/4$. (This is the only place where we use the bound on the noise in the first phase.) Thus, for any ϵ-heavy hitter u, y will favor u with probability at least $3\epsilon/4$. Also note that y can favor at most one center u, because otherwise y would have to agree with both u and u in more than half the places where they differ. So we estimate, using $O(1/\epsilon^2 \log s/\epsilon)$ samples, the fraction of y's that favor each center, and prune all but the $2/\epsilon$ centers where this estimate is greater than $\epsilon/2$.

At this point, B, the set of centers, has size less than $4/\epsilon$, where every unpruned string of T is within Hamming distance d of one of the centers. Note that there can be at most $O(n^d/\epsilon) = n^{O(\log 1/\epsilon)}$ candidates left at this point, so if noise is not independent, we can simply use this phase as our pruning algorithm to get a quasi-polynomial time enumeration algorithm.

In phase two, for each center, we enumerate those heavy hitters that are within d to that center. The union of these lists will include all heavy hitters. By taking the bit-wise parity of the noisy samples with the center in question, we can without loss of generality assume that the center in question is the all-zero string. Thus, the problem is now equivalent to enumerating all heavy hitters of small Hamming weight, i.e., d or less. For a string x, let Δ_x be the set of positions where x is 1.

Our procedure is recursive. At each point, we have two sets of positions $A, |A| \leq d$ and R, and we are trying to enumerate all low Hamming weight heavy hitters x with $A \subseteq \Delta_x \subseteq A \cup R$. Initially, A is empty, and R will be all n positions (but we show that very quickly, R will decrease to just poly$(1/\epsilon)$ positions.) We use a potential function $0 \leq Q_{A,R} \leq 1$ that allows us to prune some branches that cannot lead to actual heavy hitters. A large value of

potential function is necessary but not sufficient for there to be such a heavy hitter x with $A \subseteq \Delta_x \subseteq R \cup A$. $Q_{A,R}$ has the following properties:

1. If $|A| \leq d$, $Q_{A,R}$ can be approximated to within any poly(ϵ) additive error in time poly($1/\epsilon$).
2. If there is an ϵ-heavy hitter x of Hamming weight $\leq d$ with $A \subseteq \Delta_x \subseteq A \cup R$, then $Q_{A,R} \geq \epsilon/2$.
3. If $|R| > O(\log^2 1/\epsilon)$ and $Q_{A,R} \geq \epsilon/2$, then the average value of $Q_{A\cup\{i\},R}$ for $i \in R - A$ is at most $|R|^{-1/4} Q_{A,R}$.

Property 1 allows us to compute the potential function. Property 2 allows us to prune any branch where the potential function is too small. Property 3 says that the potential function decreases dramatically for most cases where we extend A. We first define $Q_{A,R}$ and prove it has the above properties, then describe the enumeration algorithm in terms of the properties. In the following, we use x to refer to the original sample and y to the corresponding noisy sample.

Let a and b be constants (depending on ν) so that $a\nu + b(1-\nu) = 0$, and $a(1-\nu) + b\nu = 1$ (equivalently $a = (1-\nu)/(1-2\nu)$ and $b = -\nu/(1-2\nu)$). For bit position i, let v_i be a if $y_i = 1$ and b if $y_i = 0$. Then the equations above say that the conditional expectation of v_i is x_i. (This method is borrowed from the unbiased sampler for the estimation problem from [3]. Unfortunately, using that estimator takes exponential time. We get around this by only using it within A, which has size at most $d = O(\log 1/\epsilon)$.) Let w_{R-A} be the number of 1's in y within $R-A$. Let $E_{A,R}$ be an indicator variable for the event that $w_{R-A} \leq \nu|R-A|+d-|A|$. Let $V_{A,R}$ be the random variable $(\Pi_{i\in A} v_i)E_{A,R}$. Note that, for a fixed x, all of the factors in $V_{A,R}$ are independent, so $\mathbf{E}[V_{A,R}|x] = \mathbf{P}[E_{A,R}|x]\Pi_{i\in A} x_i$. (This equation uses the fact that the noise in each bit is independent.) Let $Q_{A,R}$ be the expectation of $V_{A,R}$. The above shows that although $V_{A,R}$ may be a polynomially large positive or negative number, $Q_{A,R}$ is always between 0 and 1.

The absolute values of the v_i's are bounded by a constant, and we multiply at most d of them, so the maximum value of V is polynomial in $1/\epsilon$. Thus, we can estimate $Q_{A,R}$ up to any polynomial in ϵ additive error by averaging $V_{A,R}$ for $poly(1/\epsilon)$ samples. This establishes the first property.

If x has Hamming weight at most d, and $A \subseteq \Delta_x \subseteq R \cup A$, then if the expected number or fewer of bit flips occur in $R - A$, $E_{A,R}$ will be true. Thus, for such an x the conditional probability of $E_{A,R}$ is at least $1/2$. For such an x, $\mathbf{E}[V_{A,R}|x] \geq 1/2\Pi_{i\in A} x_i = 1/2$. Since the conditional expectation is never negative, if there is any ϵ-heavy hitter as above, $Q_{A,R} \geq \epsilon/2$. This establishes the second property.

To establish the third property, let $h_0 = (d - |A|)/(1-2\nu) + 4\sqrt{|R| \log 1/\epsilon}$. Fix any x, and let h be the Hamming weight of x in $R - A$. We claim that $\sum_{j\in R-A} \mathbf{E}[V_{A\cup\{j\},R}|x] \leq h_0(\mathbf{E}[V_{A,R}|x] + \epsilon^2)$. Note that for any $j \in R - A$, $E_{A\cup\{j\},R}$ implies $E_{A,R}$. Then

$\mathbf{E}[V_{A\cup\{j\},R}|x] = x_j(\Pi_{i\in A} x_i)\mathbf{P}[E_{A\cup\{j\},R}|x] \leq x_j(\Pi_{i\in A} x_i)\mathbf{P}[E_{A,R}|x] = x_j\mathbf{E}[V_{A,R}|x]$. Summing all $j \in |R-A|$, we get an upper bound of $h\mathbf{E}[V_{A,R}|x]$ for

the conditional expectation of the sum. So the inequality holds conditioned on any x with $h \leq h_0$.

For the other case, fix x with $h = fh_0$, $f \geq 1$. Then the expected Hamming weight of y within R is $\nu|R|+h(1-2\nu)$, and our cut-off for $E_{A,R}$ to occur is $\nu|R|+(d-|A|)$. Applying Chernoff bounds, we can upper bound the probability of $E_{A,R}$ given such an x as at most ϵ^{2f^2}. Then the overall expectation of $hE[V_{A,R}|x] \leq Ah_0\epsilon^{2f^2}$, which is decreasing with f. So setting $f = 1$ gives us an upper bound which holds in general, of $h_0\epsilon^2$ which is the second error term.

By linearity of expectation, then $\sum_j Q_{A\cup\{j\},R} \leq h_0(Q_{A,R} + \epsilon^2) \leq 2h_0Q_{A,R}$ since $Q_{A,R} \leq \epsilon/2$. So this is at most $O(\sqrt{|R-A|\log 1/\epsilon})Q_{A,R}$. Dividing by $|R-A|$ and using the assumptions $|R| \geq C'(\log 1/\epsilon)^2$ for some sufficiently large constant C' gives us that the average value of $Q_{A\cup\{j\},R}$ is at most $Q_{A,R}R^{-1/4}$. This establishes the last property.

We use this potential function $Q_{A,R}$ in our algorithm as follows: At any point, we will be enumerating those heavy hitters x with Hamming weight at most d so that $A \subset \Delta_x \subset R$ for some sets $A, |A| \leq d$ and R.

If $R < C' \log^2 1/\epsilon$, we just enumerate all d-tuples by brute force. If $Q_{A,R} < \epsilon/2$, we can just terminate and return the empty set.

Otherwise, we compute $Q_{A\cup\{j\},R}$ for each $j \in R - A$. Divide those j into the "exceptional" ones with $Q_{A\cup\{i\},R} \geq |R|^{-1/8}Q_{A,R}$ and the remaining "typical" ones. By the third property and Markov's inequality, at most $O(|R|^{7/8})$ can be exceptional. Let R' be the set of exceptional positions.

For any heavy hitter x with $A \subseteq \Delta_x \subset R$, either $\Delta_x - A \subseteq R'$ or $j \in \Delta_x$ for some $j \in R - R'$. So we output A to the list of heavy hitters, then recurse on (A, R') and recurse on $(A \cup j, R)$ for each $j \in R - R'$. This covers all of the above cases recursively.

We now have to give a time analysis for the above algorithm. Let $K = \lfloor 2Q_{A,R}/\epsilon \rfloor$ be a measure of how far we are from being able to prune our current set. Let r represent the size of R. We give our bound in terms of the number of recursive calls $T(K, r)$ needed for these values. First, if $K = 0$, we prune and terminate. So $T(0, r) = 1$. If $r < O(\log^2 1/\epsilon)$, we use brute force search and take time $(1/\epsilon)^{O(\log\log 1/\epsilon)}$. If $r > O(1/\epsilon^8)$, we make only one recursive call that isn't immediately pruned, to a subset R' of size at most $O(|R|^{7/8})$. So we shrink R without branching until $r = O(1/\epsilon^8)$. Otherwise, we make up to r "typical j" recursive calls where K decreases by a $r^{-1/8}$ factor, and one "extraordinary" recursive call where r becomes $O(r^{7/8})$. Thus, we have the recursion $T(K, r) \leq rT(Kr^{-1/8}, r) + T(2/\epsilon, r^{7/8})$. Unwinding the first part of the recursion, each time we lose a factor of $r^{1/8}$ from K, it costs us a factor of r. So we get $T(K, r) \leq K^8T(2/\epsilon, r^{7/8})$. This recursion has $O(\log\log r) = O(\log\log 1/\epsilon)$ depth, each giving a factor of $(2/\epsilon)^8$ for a total of $(1/\epsilon)^{O(\log\log 1/\epsilon)}$ recursive calls. The leaves of the recursion cost us a similar factor as mentioned earlier. During each recursion, we have to compute Q r times, which gives another polynomial factor to the total running time. Thus the total time is $(1/\epsilon)^{O(\log\log 1/\epsilon)}$, as is the list length.

4 Lower Bounds on Enumeration

We first review the result from [3] on the lower bound for the number of samples required for estimation. Let $n = \log 1/\alpha$. Consider two distributions D_0 and D_1. D_1 is the uniform distribution over n-bit strings, D_0 the uniform distribution on those n-bit strings with even parity. In D_0 the all-zero string has probability 2α, and in D_1, α, so any $\alpha/3$ estimation algorithm has to distinguish between the two.

However, both distributions are uniform when we restrict strings to any proper subset of the bit positions. Let $Q = (Q_1, Q_2, \cdots, Q_t)$ be any sequence where for $1 \leq i \leq t$, $Q_i \subset [1..n]$ is a set of bit positions. Assume that for each $1 \leq i \leq t$, i'th sample reveals exactly the bits in positions given by Q_i. Based on this condition and assuming that none of the Q_i is the entire set of positions, the induced distributions on lossy samples are identical. Any distinguishing algorithm has to wait until it sees at least one sample with all bits revealed. The probability of this occurring with any one sample is $(1-\mu)^n$, so the time required is at least $(1-\mu)^{-n} = (1-\mu)^{\log \alpha} = \alpha^{\log(1-\mu)}$. Hence if $\mu = 1 - o(1)$, this is $(1/\alpha)^{\omega(1)}$ and is super-polynomial.

Note that these two distributions give no lower bound on the number of samples required for enumeration, since an enumeration algorithm could just list all $1/\alpha$ strings of length n without seeing any samples at all. However, we will present a very similar argument that shows a lower bound for enumeration. For this, we need to increase the value of n without increasing the support size. To achieve this condition, we look at sparse distributions. We will use the fact that we are unlikely to reveal even a small constant fraction of the bit positions when $\mu = 1 - o(1)$. The following lemma and its corollary establish the existence of the desired distribution, which are stated without proof.

Lemma 1. *Let $1/2 > \gamma > 0$. There is a γn-wise independent distribution on strings of length n that has support size $2^{O(\gamma \log(\frac{1}{\gamma})n)}$.*

Corollary 1. *For any string x of length n, there is a distribution on strings of length n that is γn-wise independent, and where x has probability at least $2^{-O(\gamma \log(\frac{1}{\gamma})n)}$.*

Now, let $\mu = 1 - 1/L$. If $\mu = 1 - o(1), L = \omega(1)$. We will show that the sample complexity or the list size of any algorithm the enumerates ϵ-heavy hitters from lossy samples with erasure probability μ is super-polynomial in $1/\epsilon$. Let G be such that $L = eG^{1+\frac{G}{\log G}}$ and let $\gamma = 1/G$. G will be $\Theta(\log L)$ (since the right side is $2^{\Theta(G)}$). For any n, select a random n-bit string x and run the enumeration algorithm with $\epsilon = 2^{-\Omega(\gamma n \log \frac{1}{\gamma})} = 2^{-O(\log G/G)}$ using the γn-wise independent distribution with x in its support as obtained from the corollary. Say that the algorithm gets t lossy samples and enumerates a list of size t'. Let $S = (S_1, \cdots, S_t)$ be the sequence of sets of revealed bit positions in the samples. If every set in the sequence S has size at most γn, the observed samples are independent of x. Hence the conditional probability that x is on the

list is at most $t'/2^n$. So either $t' = \Omega(2^n) = (\frac{1}{\epsilon})^{\Omega(G/\log G)} = (\frac{1}{\epsilon})^{\Omega(\log L/\log\log L)}$ or there is a constant probability that some $|S_i| > \gamma n$. Since each position is revealed with probability $1/L$ and there are $\binom{n}{\gamma n} \leq (e/\gamma)^{\gamma n} = (eG)^{n/G}$ subsets of γn positions, the probability that any one S_i is of size $\geq \gamma n$ is at most $(eG)^{n/G}(1/L)^{n/G} = (eG/L)^{n/G}$. Hence, for any one of the S_i to have size more than γn, we must have $t > (L/eG)^{n/G} = (G^{G/\log G})^{n/G}$ by our choice of G. Now, $\epsilon = 2^{-\Omega((n\log G)/G)} = G^{-\Omega(n/G)}$, so in this case we have $t = \epsilon^{-\Omega(G/\log G} = (\frac{1}{\epsilon})^{\Omega(\log L/\log\log L)}$. Thus, if L is $\omega(1)$, then either the list size or the number of samples must be super-polynomial in $1/\epsilon$, with exponents of the form $\Omega(\log L/\log\log L)$.

The quantitative bound we get here for enumeration is almost as good as the one above for estimation. As far as we know, these are the best known lower bounds for these problems. However, they are pretty far from the upper bounds.

5 Canonical Upper and Lower Bounds for Estimation

Here, we use LP duality to show that either there is a lower bound for estimation of a certain canonical form, or an algorithm of a canonical form.

The lower bound for estimation shown in [3] has the following form: there are two distributions that differed by at least α in their probability of 0^n, the all-zero string. However, the induced distributions on lossy samples were statistically close. The distributions in [3] were completely indistinguishable unless all bits were revealed, which can only happen with small probability. Consider for now the noisy case with bit-flipping probability of ν. Any distributions D_0, D_1 that differ by α in the all-zero string must be distinguishable by an $\alpha/3$ estimation algorithm. Let N_0, N_1 be the corresponding noisy distributions, and let λ be the statistical distance of N_0 and N_1. Then any algorithm that distinguishes the two requires $t = \Omega(1/\lambda)$ samples. Now, without loss of generality, we can assume that both distributions are symmetric, so that the probabilities of outputting a Hamming weight i string for $0 \leq i \leq n$ determine them. Let Δ_i be the difference of these two probabilities for weight i. Then $\Delta_0 > \alpha$, $\sum \Delta_i = 0$ and $\sum |\Delta_i| \leq 2$. Conversely, any values of Δ_i satisfying these inequalities give rise to two such distributions whose differences are Δ_i for $0 \leq i \leq n$.

Let $m_{i,j}$ be the probability that a noisy version of a sample of Hamming weight i ends up with Hamming weight j. An explicit formula for $m_{i,j}$ is $m_{i,j} = \sum_k \binom{i}{k}(1-\mu)^k \mu^{i-k}\binom{n-i}{j-k}\mu^{j-k}(1-\mu)^{n-i-j+k}$, but we will not be using particular noisy error model right now. Our treatment works more generally for any Hamming weight to Hamming weight transformation matrix. Then the difference between N_0 and N_1's probabilities of producing a Hamming weight j string is $\sum_i m_{i,j}\Delta_i$. Thus, if we include the inequality $\sum_j |\sum_i m_{i,j}\Delta_i| \leq \lambda$ in addition to the ones above, we get that the statistical distance of the noisy versions is at most λ. Thus, these linear inequalities characterize the existence of such distributions.

In fact, these inequalities make sense in a very general setting. Let M be a stochastic $n_1 \times n_2$ metric with entries $m_{i,j}$ representing the probability: if the

original sample is of type i, that the observed sample is of type j. The above inequalities say that there are two distributions on types so that $1/\lambda$ observed samples are required to distinguish them, but differ by α in the probability of type 0 in the original distributions. So, in particular, any $\alpha/3$ estimator of the probability of type 0 requires $1/\lambda$ samples. Call such a pair of distributions (α, λ)-*fooling* pair of distributions for M.

We now consider a somewhat simplified relation of the inequalities above, but which preserve the parameters to within polynomial factors. Fix a value for λ. The primal relaxation has objective to maximize Δ_0 subject to the following inequalities in variables $\Delta_0, \ldots, \Delta_n$: 1. $\sum_{i=0}^{i=n} \Delta_i = 0$, 2. For each $0 \leq i \leq n_1$, $-1 \leq \Delta_i \leq 1$, and 3. For each $0 \leq j \leq n_2, -\lambda \leq \sum m_{i,j}\Delta_i \leq \lambda$

If there is a solution with $\Delta_0 \geq \alpha$, we can give two distributions N_0 and N_1 as follows. Let R be the sum of the positive Δ_i. Note that $n_1 + 1 \geq \sum_i |\Delta_i| \geq R \geq \Delta_0 \geq \alpha$. N_0 is supported on those i with $\Delta_i > 0$, and the probability of strings with Hamming weight i solutions is Δ_i/R for such i. N_1 is supported on those i with $\Delta_i < 0$, and the probability of Hamming weight i is $-\Delta_i/R$.

For any i, the difference between the probabilities is Δ_i/R. Thus, the difference between their probabilities of the all-zero string is $\Delta_0/R \geq \alpha/(n_1 + 1)$, and the statistical distance between the noisy versions is: $\sum_j |\sum_i m_{i,j}\Delta_i/R| = 1/R\sum_j |\sum_i m_{i,j}\Delta_i| \leq \lambda(n_2 + 1)/R \leq \lambda(n_2 + 1)/\alpha$.

So if the optimum objective is greater than α, there are two distributions such that their noisy versions are indistinguishable to within $\lambda(n_2 + 1)/\alpha$. Thus, any $\alpha/(3(n_1 + 1))$ estimation algorithm will require $\alpha/((n_2 + 1)\lambda)$ samples.

If the optimum objective is less than α, consider the dual system of inequalities. Say we multiply the first equation by $w > 0$, the lower bound in the i'th example of the second set of inequalities by $u_i > 0$, the upper bound by $v_i > 0$, the lower bound in the j'th example in the third set of inequalities by $y_j > 0$ and the upper bound by $z_j > 0$. Then we get the induced inequality

$\sum_i (w + v_i - u_i + \sum_j (z_j - y_j)m_{i,j})\Delta_i \leq \sum_i (u_i + v_i) + \lambda \sum_j (y_j + z_j)$.

So the dual is to minimize $\sum_i (u_i + v_i) + \lambda \sum_j (y_j + z_j)$ subject to $w + v_0 - u_0 + \sum_j (z_j - y_j)m_{0,j} = 1$ and $w + v_i - u_i + \sum_j (z_j - y_j)m_{i,j} = 0$ for $n_1 \geq i \geq 1$. Note that if the objective is less than α, then so is each u_i and v_i, each y_j and z_j are at most α/λ, and $w < |v_1 - u_1| + max_j |z_j - y_j| = O(\alpha/\lambda)$.

Let $\beta_j = (z_j - y_j) + w$, and consider the algorithm : Let J be the random variable given by the observed type of the noisy sample. Estimate the expected value of β_J to within $O(\alpha)$. The constraints say that, on a type 0 input, the expectation of β_J is $1+O(\alpha)$ and on any other Hamming weight, the expectation is $O(\alpha)$ in absolute value.

Thus, the expectation of β_J will be within $O(\alpha)$ of the probability of the original sample being type 0. Since each $|\beta_j|$ is at most $O(\alpha/\lambda)$, we can estimate the expectation to within α using $O(1/\lambda^2)$ samples. In the language of [3]. the vector $\boldsymbol{\beta}$ is a *local inverse* of M at 0. We summarize the discussion in the following:

Theorem 1. *Let M be any $n_1 \times n_2$ stochastic matrix, and let $0 < \alpha, \lambda < 1$. Then either there is an pair of (α, λ)-fooling distributions for M (and hence, any $\alpha/3$-estimator for the probability of 0 requires $\Omega(1/\lambda)$ samples) or there is*

a local inverse for M with maximum coefficients of size $O(n_1 n_2 \alpha/\lambda)$ (and hence there is a time polynomial in n_1, n_2 and $1/\lambda$ algorithm to estimate the probability of 0 within $O(\alpha)$).

Acknowledgments. We are grateful to Avi Wigderson for introducing us to the problem and for useful discussions and suggestions, and to Amir Yehudayoff for helpful comments on an earlier version. We thank Chris Beck, Shachar Lovett, Mike Saks, and Ankur Moitra for helpful conversations.

References

1. Belkin, M., Sinha, K.: Polynomial learning of distribution families. In: FOCS 2010, pp. 103–112 (2010)
2. Dasgupta, S.: Learning mixtures of gaussians. In: FOCS 1999, p. 634. Computer Society (1999)
3. Dvir, Z., Rao, A., Wigderson, A., Yehudayoff, A.: Restriction access. In: Innovations in Computer Science 2012, pp. 19–33 (2012)
4. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: STOC 1989, pp. 25–32 (1989)
5. Karp, R.M., Shenker, S., Papadimitriou, C.H.: A simple algorithm for finding frequent elements in streams and bags. ACM Trans. Database Syst. 28(1), 51–55 (2003)
6. Kearns, M., Mansour, Y., Ron, D., Rubinfeld, R., Schapire, R.E., Sellie, L.: On the learnability of discrete distributions. In: STOC 1994, pp. 273–282 (1994)
7. Moitra, A., Saks, M.: A polynomial time algorithm for lossy population recovery. Manuscript (2013)
8. Moitra, A., Valiant, G.: Settling the polynomial learnability of mixtures of gaussians. In: FOCS 2010, pp. 93–102 (2010)
9. Arora, S., Kannan, R.: Learning mixtures of arbitrary gaussians. In: STOC 2001, pp. 247–257 (2001)
10. Wigderson, A., Yehudayoff, A.: Population recovery and partial identification. In: FOCS 2012, pp. 390–399 (2012)
11. Woeginger, G.J.: When does a dynamic programming formulation guarantee the existence of an fptas? In: SODA 1999, pp. 820–829 (1999)

Private Learning and Sanitization: Pure vs. Approximate Differential Privacy*

Amos Beimel[1], Kobbi Nissim[2], and Uri Stemmer[1]

[1] Dept. of Computer Science Ben-Gurion University
[2] Dept. of Computer Science Ben-Gurion University & Harvard University
{beimel,kobbi,stemmer}@cs.bgu.ac.il

Abstract. We compare the sample complexity of private learning and sanitization tasks under *pure* ϵ-differential privacy [Dwork, McSherry, Nissim, and Smith TCC 2006] and *approximate* (ϵ, δ)-differential privacy [Dwork, Kenthapadi, McSherry, Mironov, and Naor EUROCRYPT 2006]. We show that the sample complexity of these tasks under approximate differential privacy can be significantly lower than that under pure differential privacy.

Keywords: Differential Privacy, Private Learning, Sanitization.

1 Introduction

Learning and sanitization are often applied to collections of sensitive data of individuals and it is important to protect the privacy of these individuals. We examine the sample complexity of these tasks while preserving differential privacy [9]. Our main focus is on private learning [14] and sanitization [4] and we show starking differences between the required sample complexity for these tasks under ϵ-differential privacy [9] (also called *pure* differential privacy) and its variant (ϵ, δ)-differential privacy [7] (also called *approximate* differential privacy).

An algorithm A satisfies the requirement of *Pure Differential Privacy* if for every two databases that differ on exactly one entry, and for every event defined over the output of the algorithm, the probability of this event is close up to a multiplicative factor of $\approx 1 + \epsilon$ whether A is applied on one database or on the other. Informally, to satisfy the requirement of *Approximate Differential Privacy* the above guarantees needs to be satisfied only for events whose probability is at least $\approx \delta$. We show that even negligible $\delta > 0$ can have a significant effect on sample complexity of two fundamental tasks: private learning and sanitization.

* Research supported by the Israel Science Foundation (grants No. 938/09 and 2761/12) and by the Frankel Center for Computer Science at Ben-Gurion University. Work done while the second author was a Visiting Scholar at the Harvard Center for Research on Computation and Society (CRCS). Work partially done when the third author was visiting Harvard University supported in part by NSF grant CNS-1237235 and a gift from Google, Inc.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 363–378, 2013.

Private Learning. Private learning was introduced in [14] as a combination of Valiant's PAC learning model [21] and differential privacy (applied to the examples used by the learner). The work on private learning has since mainly focused on pure privacy. On the one hand, this work showed, via generic constructions [3,14], that every finite concept $\mathcal{C}$ class can be learned privately, using sample complexity proportional to $\text{poly}(\log|\mathcal{C}|)$ (often efficiently). On the other hand, a significant difference was shown between the sample complexity of *traditional* (non-private) learners (crystallized in terms of $\text{VC}(\mathcal{C})$ and smaller than $\log|\mathcal{C}|$ in many interesting cases) and private learners, when the latter are required to be *proper* (i.e., output a hypothesis in $\mathcal{C}$).

As an example, let $\texttt{POINT}_d$ be the class of point functions over the domain $\{0,1\}^d$ (these are the functions that evaluate to one on exactly one point of the domain). Consider the task of *properly* learning $\texttt{POINT}_d$ where, after consulting its sample, the learner outputs a hypothesis that is by itself in $\texttt{POINT}_d$. Non-privately, learning $\texttt{POINT}_d$ requires merely a constant number of examples (as $\text{VC}(\texttt{POINT}_d) = 1$). Privately, $\Omega(d)$ examples are required [1]. Curiously, the picture changes when the private learner is allowed to output a hypothesis not in $\texttt{POINT}_d$ (such learners are called *improper*), as the sample complexity can be reduced to $O(1)$ [1]. This, however, comes with a price, as it was shown [1] that such learners must return hypotheses that evaluate to one on exponentially many points in $\{0,1\}^d$ and, hence, are very far from all functions in $\texttt{POINT}_d$. A similar lower bound of $\Omega(d)$ samples is known also for properly and privately leaning the class $\texttt{INTERVAL}_d$ of threshold functions over the interval $[0, 2^d - 1]$ (no corresponding sample-efficient improper private learner is known, that is, the best previously known private learning algorithm (proper or improper) for $\texttt{INTERVAL}_d$ has sample complexity $O(d)$). A complete characterization for the sample complexity of pure private learners was recently given in [2], in terms of a new dimension – the *Representation Dimension*, that is, given a class C the number of samples needed and sufficient for privately learning C is $\text{RepDim}(C)$.

We show that the sample complexity of proper private learning with approximate differential privacy can be significantly lower than that with pure differential privacy. Our starting point for this work is an observation that with *approximate* $(\epsilon, \delta{>}0)$-differential privacy, sample complexity of $O(\log(1/\delta))$ suffices for learning points *properly*. This gives a separation between pure and approximate proper private learning for $\delta = 2^{-o(d)}$. Anecdotally, combining this with a result from [1] gives a learning task that is not computationally feasible under pure differential privacy and polynomial time computable under approximate differential privacy.

Sanitization. The notion of differentially private sanitization was introduced in the seminal work of Blum et al. [4]. A sanitizer for a class of predicates $\mathcal{C}$ is a differentially private mechanism translating an input database S to an output database $\hat{S}$ s.t. $S, \hat{S}$ agree (approximately) on the fraction of the entries in S satisfying φ for all $\varphi \in \mathcal{C}$. Blum et al. gave a generic construction of pure differentially private sanitizers exhibiting sample complexity $O(\text{VC}(C)\log|X|)$, and lower bounds partially supporting this sample complexity were given by [17,1,13]

(the construction is not generally feasible [10,20,19]). As with private learning, we show significant differences between the sample complexity required for sanitization of simple predicate classes under pure and approximate differential privacy.

1.1 Our Contributions

To simplify the exposition, we omit in this section dependency on all variables except for the complexity variable d corresponding, e.g., to domain size and (logarithm of) concept class size.

Tools. A recent instantiation of the Propose-Test-Release framework [8] by Smith and Thakurta [18] results, almost immediately, with a proper learner for points, exhibiting $O(1)$ sample complexity while preserving approximate differential privacy. This simple technique does not suffice for our other constructions of learners and sanitizers, and we, hence, introduce new tools for coping with proper private learning of intervals, sanitization for point functions, and sanitization for intervals:

- **Choosing mechanism:** Given a *low-sensitivity* quality function, one can use the exponential mechanism [16] to choose an approximate solution. This method requires, in general, a database of size logarithmic in the number of possible solutions. We identify a sub family of low-sensitivity functions, called *bounded-growth* functions, for which it is possible to significantly reduce the necessary database size when using the exponential mechanism.
- **Recursive algorithm for concave promise problems:** We define a family of optimization problems, which we call *Concave Promise Problems*. The possible solutions are ordered, and *concavity* means that if two solutions $f \leq h$ have quality $\geq \mathcal{X}$, then any solution $f \leq g \leq h$ also has quality $\geq \mathcal{X}$. The optimization goal is, when there exists a solution with a promised quality $\geq r$, to find a solution with quality $\approx r$. We observe that a concave promise problem can be privately approximated using a solution to a smaller instance of a concave promise problem. This allows us to construct an efficient recursive algorithm solving such problems privately. We show that the task of learning $\texttt{INTERVAL}_d$ is, in fact, a concave promise problem, and can be privately solved using our algorithm with sample size roughly $2^{\log^* d}$. Sanitization for $\texttt{INTERVAL}_d$ does not exactly fit the model of concave promise problems but can still be solved by iteratively defining and solving a small number of concave promise problems.

Implications for Private Learning and Sanitization. We give new *proper* private learning algorithms for the classes $\texttt{POINT}_d$ and $\texttt{INTERVAL}_d$. Both algorithms exhibit sample complexity that is significantly lower than bounds given in prior work, separating pure and approximate proper private learning. Similarly, we construct sanitizers for these two classes, again with sample complexity that is significantly lower than bounds given in prior work, separating sanitization in the pure and approximate privacy cases.

Santization vs. Private Learning. In [11], a reduction is given in both directions between agnostic learning of a concept class C, and the sanitization task for the same class C. They consider learners and sanitizers with limited access to the data, using statistical queries [15] (a non-private SQ learner could be transformed into a private learner, as was shown in [14]). In Section 5 we show a different (and natural) reduction from the task of privately learning a concept class C to the sanitization task of a slightly different concept class C', where the sanitizer's access to the database is unrestricted. We then exploit lower bounds on the sample complexity of private learners and show a class of predicates $\mathcal{C}$ over a domain X for which every private sanitizer requires databases of size $\Omega(\mathrm{VC}(\mathcal{C}) \log |X|)$. A similar lower bound was already shown by Hardt and Rothblum [13], achieving better results in terms of the approximation parameter. Their work ensures the existence of such a concept class, but does not give an explicit one.

Label Privacy. In Section 6 we examine private learning under a relaxation of differential privacy called *label privacy* (see [5] and references therein) where the learner is required to only protect the privacy of the labels in the sample. Chaudhuri et al. [5] gave lower bounds for label-private learners in terms of the doubling dimension of the target concept class. We show that the VC dimension completely characterizes the sample complexity of such learners.

1.2 Related Work

Mostly related to our work is the work on private learning and its sample complexity [14,3,1,5] and the early work on sanitization [4] mentioned above. Another related work is the work of De [6] who proved a separation between *pure* ϵ-differential privacy and *approximate* $(\epsilon, \delta{>}0)$-differential privacy. Specifically, he demonstrated that there exists a query where it is sufficient to add noise $O(\sqrt{n \log(1/\delta)})$ when $\delta > 0$ and $\Omega(n)$ noise is required when $\delta = 0$. Earlier work by Hardt and Talwar separated pure from approximate differential privacy for $\delta = n^{-O(1)}$ [12].

2 Preliminaries

Notation. We use $O_\gamma(g(d))$ as a shorthand for $O(h(\gamma) \cdot g(d))$ for some non-negative function h. We use X to denote an arbitrary domain, and X_d for the domain $\{0,1\}^d$. Databases $S_1 \in X^m$ and $S_2 \in X^m$ over a domain X are called *neighboring* if they differ in exactly one entry.

2.1 Differential Privacy

Definition 1 ([9,7]). *A randomized algorithm A is (ϵ, δ)-differentially private if for all neighboring databases S_1, S_2, and for all sets $\mathcal{F}$ of outputs,* $\Pr_A[A(S_1) \in \mathcal{F}] \leq \exp(\epsilon) \cdot \Pr_A[A(S_2) \in \mathcal{F}] + \delta$. *When $\delta = 0$ we omit it and say that A preserves ϵ-differential privacy.*

We use the term *pure* differential privacy when $\delta = 0$ and the term *approximate* differential privacy when $\delta > 0$.

2.2 PAC Learning and Private PAC Learning

A concept $c : X \to \{0,1\}$ is a predicate that labels *examples* taken from the domain X by either 0 or 1. A *concept class* $\mathcal{C}$ over X is a set of concepts mapping X to $\{0,1\}$. A learning algorithm is given examples sampled according to an unknown probability distribution $\mathcal{D}$ over X, and labeled according to an unknown *target* concept $c \in \mathcal{C}$. The goal of the learning algorithm is to output a hypothesis h that approximates c well over samples from $\mathcal{D}$.

Definition 2. *The* generalization error *of a hypothesis* $h : X \to \{0,1\}$ *is defined as* $\text{error}_{\mathcal{D}}(c,h) = \Pr_{x\sim\mathcal{D}}[h(x) \neq c(x)]$. *If* $\text{error}_{\mathcal{D}}(c,h) \leq \alpha$ *we say that* h *is* α-good *for* c *and* $\mathcal{D}$.

Definition 3 (PAC Learning [21]). *Algorithm* A *is an* (α,β,m)-PAC learner *for a concept class* $\mathcal{C}$ *over* X *using hypothesis class* $\mathcal{H}$ *if for all concepts* $c \in \mathcal{C}$, *all distributions* $\mathcal{D}$ *on* X, *given an input of* m *samples* $S = (z_1,\ldots,z_m)$, *where* $z_i = (x_i, c(x_i))$ *and* x_i *are drawn i.i.d. from* $\mathcal{D}$, *algorithm* A *outputs a hypothesis* $h \in \mathcal{H}$ *satisfying* $\Pr[\text{error}_{\mathcal{D}}(c,h) \leq \alpha] \geq 1-\beta$. *The probability is taken over the random choice of the examples in* S *according to* $\mathcal{D}$ *and the coin tosses of the learner* A. *If* $\mathcal{H} \subseteq \mathcal{C}$ *then* A *is called a* proper *PAC learner; otherwise, it is called an* improper *PAC learner.*

Definition 4. *For a labeled sample* $S = (x_i, y_i)_{i=1}^m$, *the* empirical error *of* h *w.r.t.* S *is* $\text{error}_S(h) = \frac{1}{m}|\{i : h(x_i) \neq y_i\}|$.

In private learning, we would like to accomplish the same goal as in non-private learning, while protecting the privacy of the input database.

Definition 5 (Private PAC Learning [14]). *Let* A *be an algorithm that gets an input* $S = \{z_1,\ldots,z_m\}$. *Algorithm* A *is an* $(\alpha,\beta,\epsilon,\delta,m)$-PPAC learner *for a concept class* $\mathcal{C}$ *over* X *using hypothesis class* $\mathcal{H}$ *if (i) Algorithm* A *is* (ϵ,δ)-*differentially private; and (ii) Algorithm* A *is an* (α,β,m)-PAC learner *for* $\mathcal{C}$ *using* $\mathcal{H}$. *When* $\delta = 0$ *(pure privacy) we omit it from the list of parameters.*

2.3 Private Data Release

Given a concept $c : X \to \{0,1\}$, the counting query $Q_c : X^* \to [0,1]$ is defined as $Q_c(S) = \frac{1}{|S|} \cdot |\{x_i \in S : c(x_i) = 1\}|$. That is, $Q_c(S)$ is the fraction of entries in S that satisfy the concept c. Given a database S, a sanitizer for a concept class C is required to approximate $Q_c(S)$ for every $c \in C$.

Definition 6 (Sanitization [4]). *Let* $\mathcal{C}$ *be a class of concepts mapping* X *to* $\{0,1\}$. *Let* A *be an algorithm that on an input database* $S \in X^*$ *outputs a description of a function* $\text{est} : \mathcal{C} \to [0,1]$. *Algorithm* A *is an* $(\alpha,\beta,\epsilon,\delta,m)$-*improper-sanitizer for predicates in the class* $\mathcal{C}$, *if (i)* A *is* (ϵ,δ)-*differentially private; and, (ii) For every input* $S \in X^m$, $\Pr_A\left[\forall c \in C \ \left|Q_c(S) - \text{est}(c)\right| \leq \alpha\right] \geq 1-\beta$.

If on an input database S algorithm A outputs another database $\hat{S} \in X^*$, and $\text{est}(\cdot)$ *is defined as* $\text{est}(c) = Q_c(\hat{S})$*, then algorithm A is called a* proper-sanitizer *(or simply a* sanitizer*). As before, when $\delta = 0$ (pure privacy) we omit it from the set of parameters. A database S and a function* est *(or two databases $S, \hat{S}$) are called* α-close *if $|Q_c(S) - \text{est}(c)| \leq \alpha$ for every $c \in C$.*

Ignoring computational complexity, an $(\alpha, \beta, \epsilon, \delta, m)$-improper-sanitizer can always be transformed into a $(2\alpha, \beta, \epsilon, \delta, m)$-sanitizer, by finding a database $\hat{S}$ of m entries that is α-close to the returned estimation est (such a database exists, i.e., S itself).

2.4 Basic Differentially Private Mechanisms

The Laplace Mechanism. The most basic construction of differentially private algorithm is via the Laplace mechanism as follows.

Definition 7 (The Laplace Distribution). *A random variable has probability distribution* $\text{Lap}(b)$ *if its probability density function is* $f(x) = \frac{1}{2b}\exp(-\frac{|x|}{b})$.

Definition 8 (Sensitivity). *A function $f : X^m \to \mathbb{R}$ has* sensitivity k *if for every neighboring $D, D' \in X^m$, it holds that $|f(D) - f(D')| \leq k$.*

Theorem 1 (The Laplacian Mechanism [9]). *Let $f : X^m \to \mathbb{R}$ be a sensitivity k function. The mechanism A which on input $D \in X^m$ outputs $A(D) = f(D) + \text{Lap}(\frac{k}{\epsilon})$ preserves ϵ-differential privacy. Moreover,* $\Pr[|A(D) - f(D)| > \Delta] = \exp\left(-\frac{\epsilon\Delta}{k}\right)$.

The Exponential Mechanism. Let X be a domain and $\mathcal{H}$ a set of solutions. Given a quality function $q : X^* \times \mathcal{H} \to \mathbb{N}$, and a database $S \in X^*$, the goal is to chooses a solution $h \in \mathcal{H}$ approximately maximizing $q(S, h)$. The exponential mechanism, by McSherry and Talwar [16], chooses a solution $h \in H$ with probability proportional to $\exp(\epsilon \cdot q(S, h)/2)$.

Proposition 1 (Properties of the Exponential Mechanism). *(i) The exponential mechanism is ϵ-differentially private. (ii) Let $\hat{e} \triangleq \max_{f \in \mathcal{H}}\{q(S, f)\}$. The exponential mechanism outputs a solution h such that $q(S, h) \leq (\hat{e} - \Delta m)$ with probability at most $|\mathcal{H}| \cdot \exp(-\epsilon\Delta m/2)$.*

Stability and Privacy. We restate a simplified variant of algorithm $\mathcal{A}_{dist}$ by Smith and Thakurta [18], which is an instantiation of the PTR framework [8]. Let $q : X^* \times F \to \mathbb{N}$ be a sensitivity-1 quality function over a domain X and a set of solutions F. Given a database $S \in X^*$, the goal is to chooses a solution $f \in F$ maximizing $q(S, f)$, under the assumption that the optimal solution f scores much better than any other solution in F.

Algorithm $\mathcal{A}_{dist}$
Input: parameters ϵ, δ, database $S \in X^*$, sensitivity-1 quality function q.
1. Let $f_1 \neq f_2$ be two highest score solutions in F, where $q(f_1, S) \geq q(f_2, S)$.
2. Let $\text{gap} = q(f_1, S) - q(f_2, S)$ and $\text{gap}^* = \text{gap} + \text{Lap}(\frac{4}{\epsilon})$.
3. If $\text{gap}^* < \frac{4}{\epsilon}\ln(\frac{1}{\delta}) + 2$ then output $\perp_1$ and halt.
4. If $\text{gap} = 0$ then output $\perp_2$, otherwise output f_1.

Proposition 2 (Properties of $\mathcal{A}_{dist}$ [18]). *(i) Algorithm $\mathcal{A}_{dist}$ is (ϵ, δ)- differentially private. (ii) When given an input database S for which $\text{gap} \geq \frac{4}{\epsilon}\ln(\frac{2}{\beta\delta})$, algorithm $\mathcal{A}_{dist}$ outputs f_1 maximizing $q(f, S)$ with probability at least $(1 - \beta)$.*

3 Learning with Approximate Privacy

We present proper (ϵ, δ)-private learners for two simple concept classes, $\texttt{POINT}_d$ and $\texttt{INTERVAL}_d$, demonstrating separations between pure and approximate private proper learning.

3.1 (ϵ, δ)-PPAC Learner for $\texttt{POINT}_d$

For $j \in X_d$ let $c_j : X_d \to \{0, 1\}$ be defined as $c_j(x) = 1$ if $x = j$ and $c_j(x) = 0$ otherwise. Define the concept class $\texttt{POINT}_d = \{c_j\}_{j \in X_d}$. Note that the VC dimension of $\texttt{POINT}_d$ is 1, and, therefore, there exists a *proper* non-private learner for $\texttt{POINT}_d$ with sample complexity $O_{\alpha,\beta}(1)$. Beimel at el. [1] proved that every *proper* ϵ-private learner for $\texttt{POINT}_d$ must have sample complexity $\Omega(d) = \Omega(\log|\texttt{POINT}_d|)$. They also showed that there exists an *improper* ϵ-private learner for this class, with sample complexity $O_{\alpha,\beta,\epsilon}(1)$.

As we will now see, algorithm $\mathcal{A}_{dist}$ (defined in Section 2.4) can be used as a *proper* (ϵ, δ)-private learner for $\texttt{POINT}_d$ with sample complexity $O_{\alpha,\beta,\epsilon,\delta}(1)$. This is our first (and simplest) example separating the sample complexity of pure and approximate private learners. Consider the following algorithm.

Input: parameters $\alpha, \beta, \epsilon, \delta$, and a database $S \in X_{d+1}^m$.
1. For every x, define $q(S, x)$ as the number of appearances of $(x, 1)$ in S.
2. Execute $\mathcal{A}_{dist}$ on S with the quality function q and parameters $\frac{\alpha}{2}, \frac{\beta}{2}, \epsilon, \delta$.
3. If the output was j then return c_j.
4. Else, if the output was $\perp_1$ or $\perp_2$ then return a random $c_i \in \texttt{POINT}_d$.

Lemma 1. *Let $\alpha, \beta, \epsilon, \delta$ be s.t. $\frac{1}{\alpha\beta} \leq 2^d$. The above algorithm is an efficient $(\alpha, \beta, \epsilon, \delta)$-PPAC proper learner for $\texttt{POINT}_d$ using a sample of $m = O\left(\frac{1}{\alpha\epsilon}\ln(\frac{1}{\beta\delta})\right)$ labeled examples.*

The proof is omitted from this extended abstract. For intuition, consider a target concept c_j and an underling distribution $\mathcal{D}$. Whenever $\mathcal{D}(j)$ is noticeable, a

typical sample S contains many copies of the point j labeled as 1. As every other point $i \neq j$ will be labeled as 0, we expect $q(S, j)$ to be significantly higher than any other $q(S, i)$, and we can use algorithm $\mathcal{A}_{dist}$ to identify j.

3.2 Towards a Proper (ϵ, δ)-PPAC Learner for $\texttt{INTERVAL}_d$

For $0 \leq j \leq 2^d$ let $c_j : X_d \to \{0, 1\}$ be defined as $c_j(x) = 1$ if $x < j$ and $c_j(x) = 0$ otherwise. Define the concept class $\texttt{INTERVAL}_d = \{c_j\}_{0 \leq j \leq 2^d}$. Note that $\mathrm{VC}(\texttt{INTERVAL}_d) = 1$, and, therefore, there exists a *proper* non-private learner for $\texttt{INTERVAL}_d$ with sample complexity $O_{\alpha,\beta}(1)$. As $|\texttt{INTERVAL}_d| = 2^d + 1$, one can use the generic construction of Kasiviswanathan et al. [14] and get a proper ϵ-private learner for this class with sample complexity $O(d)$. Beimel et al. [1] showed that this is in fact optimal, and every proper ϵ-private learner for this class must have sample complexity $\Omega(d)$. It is unknown whether there exists an improper ϵ-private learner for $\texttt{INTERVAL}_d$ with sample complexity $o(d)$.

Our learner for $\texttt{POINT}_d$ relied on a strong "stability" property of the problem: Given a labeled sample, either a random concept is (w.h.p.) a good output, or, there is exactly one consistent concept in the class, and every other concept has large empirical error. This, however, is not the case when dealing with $\texttt{INTERVAL}_d$. In particular, many hypotheses can have low empirical error, and changing a single entry of a sample S can significantly affect the set of hypotheses consistent with it.

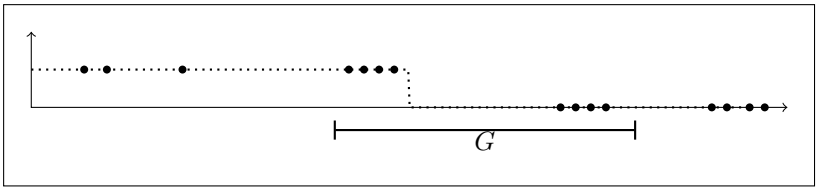
In Section 3.3, we present a proper (ϵ, δ)-private learner for $\texttt{INTERVAL}_d$ exhibiting sample complexity $\widetilde{O}_{\alpha,\beta,\epsilon,\delta}(16^{\log^*(d)})$. We use this section for motivating the construction. We start with two simplifying assumptions. First, when given a labeled sample S, we aim at choosing a hypothesis $h \in \texttt{INTERVAL}_d$ approximately minimizing the empirical error (rather than the generalization error). Second, we assume that we are given a "diverse" sample S that contains many points labeled as 1 and many points labeled as 0. Those two assumptions (and any other informalities made hereafter) will be removed in Section 3.3.

Assume we are given as input a sample $S = (x_i, y_i)_{i=1}^m$ labeled by some unknown $c_\ell \in \texttt{INTERVAL}_d$. We would now like to choose a hypothesis $h \in \texttt{INTERVAL}_d$ with small empirical error on S, and we would like to do so while accessing the sample S only through differentially private tools.

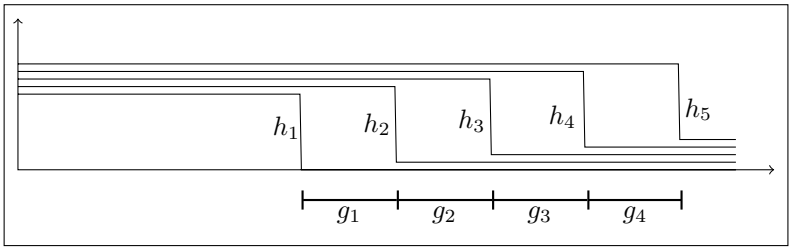
We will refer to points labeled as 1 in S as *ones*, and to points labeled as 0 as *zeros*. Imagine for a moment that we already have a differentially private algorithm that given S outputs an interval $G \subseteq X_d$ with the following two properties:

1. The interval G contains "a lot" of ones, *and* "a lot" of zeros in S.
2. Every interval $I \subseteq X_d$ of length $\leq \frac{|G|}{k}$ does not contain, simultaneously, "too many" ones and "too many" zeros in S, where k is some constant.

Such an interval will be referred to as a *k-good* interval. The figure below illustrates such an interval G, where the dotted line represents the (unknown) target concept, and the bold dots correspond to sample points.

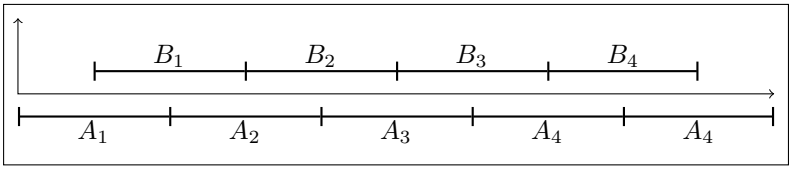


Given such a 4-good interval G, we can (without using the sample S) define a set H of five hypotheses, s.t. at least one of them has small empirical error. To see this, consider the figure below, where G is divided into four equal intervals g_1, g_2, g_3, g_4, and 5 hypotheses $h_1, \ldots, h_5$ are defined s.t. the points where they switch from one to zero are uniformly spread inside G. Now, as the interval G contains both ones and zeros, it must be that the target concept c_ℓ switches from 1 to 0 inside G. Assume without loss of generality that this switch occurs inside g_2. Note that g_2 is of length $\frac{|G|}{4}$ and, therefore, either does not contain too many ones, and h_2 is "close" to the target concept, or does not contain too many zeros, and h_3 is "close" to the target concept.



After defining such a set H, we could use the exponential mechanism to choose a hypothesis $h \in H$ with small empirical error on S. As the size of H is constant, this requires only a constant number of samples. To conclude, finding a k-good interval G (while preserving privacy) is sufficient for choosing a good hypothesis. We next explain how to find such an interval.

Assume, for now, that we have a differentially private algorithm that given a sample S, returns an *interval length* J s.t. there exists a 2-good interval $G \subseteq X_d$ of length $|G| = J$. This length J could be used to find an explicit 4-good interval as follows. Divide X_d into intervals $\{A_i\}$ of length $2J$, and into intervals $\{B_i\}$ of length $2J$ right shifted by J as in the figure below.



As the promised 2-good interval G is of length J, at least one of the above intervals contains G. If, e.g., $G \subseteq A_2$ then A_2 contains both a lot of zeros and a lot of ones. The target concept must switch inside A_2, and, therefore, every

other $A_i \neq A_2$ cannot contain both zeros and ones. For every interval A_i, define its quality $q(A_i)$ to be the minimum between the number of zeros in A_i and the number of ones in A_i. We have, therefore, that $q(A_2)$ is big, while $q(A_i) = 0$ for every $A_i \neq A_2$. That is, A_2 scores much better than any other A_i under this quality function q. The sensitivity of $q()$ is one and we can use algorithm $\mathcal{A}_{dist}$ to privately identify A_2. Recall that $G \subseteq A_2$ is a 2-good interval, and that $|A_2| = 2|G|$. The identified A_2 is, therefore, a 4-good interval.

To conclude, if we could indeed find (while preserving privacy) a length J s.t. there exists a k-good interval G of that length – our task would be completed.

Computing the Interval Length J. At first attempt, one might consider preforming a binary search for such a length $0 \leq J \leq 2^d$, in which every comparison will be made using the Laplace mechanism. This will indeed preserve privacy. However, as there are d noisy comparisons, this solution will require a sample of size $d^{\Omega(1)}$ in order to achieve reasonable utility guarantees.

As a second attempt, one might consider preforming a binary search, not on $0 \leq J \leq 2^d$, but rather on the power j of an interval of length 2^j. That is, preforming a search for a power $0 \leq j \leq d$ for which there exists a 2-good interval of length 2^j. Here there are only $\log(d)$ noisy comparisons, and the sample size will be reduced to $\log^{\Omega(1)}(d)$. More specifically, for every power $0 \leq j \leq d$, define

$$Q(j) = \max_{\substack{[a,b] \subseteq X_d \\ b-a=2^j}} \left\{ \min \left\{ \begin{array}{c} \text{number of} \\ \text{zeros in } [a,b] \end{array}, \begin{array}{c} \text{number of} \\ \text{ones in } [a,b] \end{array} \right\} \right\}.$$

If, e.g., we have that $Q(j) = 100$ for some j, then *there exists* an interval $[a,b] \subseteq X_d$ of length 2^j that contains at least 100 ones and at least 100 zeros. Moreover, *every* interval of length $\leq 2^j$ either contains at most 100 ones, or, contains at most 100 zeros.

A binary search on $0 \leq j \leq d$ can (privately) yield an appropriate length $J = 2^j$ s.t. $Q(j)$ is "big enough" (and so, there exists an interval of length 2^j containing lots of ones and lots of zeros), while $Q(j-1)$ is "small enough" (and so, every interval of length $\frac{1}{2}2^j$ can not contain too many ones and too many zeros simultaneously).

Remark 1. A binary search as above would have to operate on noisy values of $Q(\cdot)$ (as otherwise differential privacy cannot be obtained). For this reason we set the bounds for "big enough" and "small enough" to overlap. Namely, we search for a value j such that $Q(j) \geq (1 - \frac{3\alpha}{4})m$ and $Q(j-1) \leq (1 - \frac{\alpha}{4})m$, where α is our approximation parameter, and m is the sample size.

We will apply recursion to reduce the costs of computing $J = 2^j$ to $2^{O(\log^*(d))}$. The tool performing the recursion would be formalized and analyzed in the next section. This tool will later be used in our construction of a proper (ϵ, δ)-private learner for $\texttt{INTERVAL}_d$.

3.3 Privately Approximating Concave Promise Problems

Definition 9. *A function $Q(\cdot)$ is* concave *if $Q(i), Q(j) \geq x$ implies $Q(\ell) \geq x$ for every $i \leq \ell \leq j$.*

Definition 10 (Concave Promise Problem). *A* Concave Promise Problem *consists of an interval of possible solutions $[0,T] = \{0,1,\ldots,T\}$, a database $S \in X^m$, a sensitivity-1 quality function $Q : X^* \times [0,T] \to \mathbb{R}$, an approximation parameter α, and another parameter r (called a* quality promise*).*

If $Q(S,\cdot)$ is concave and if there exists an solution $p \in [0,T]$ for which $Q(S,p) \geq r$ then a good output for the problem is a solution $k \in [0,T]$ satisfying $Q(S,k) \geq (1-\alpha)r$. The outcome is not restricted otherwise.

We are interested in solving concave promise problems while preserving differential privacy (privacy must be preserved even when $Q(S,\cdot)$ is not concave or $Q(S,p) < r$ for all $p \in [0,T]$). Our algorithm *Rec* is presented in Fig. 1 (see inline comments for some of the underlying intuition).

Lemma 2. *When executed on a range $[0,T]$, a sensitivity-1 quality function Q, and parameters ϵ, δ, algorithm Rec preserves $(\epsilon', 4\log^*(T)\delta)$-differential privacy, where $\epsilon' = \sqrt{6\log^*(T)\ln(\frac{1}{\log^*(T)\delta})} \cdot \epsilon + 6\log^*(T) \cdot \epsilon^2$.*

Lemma 3. *Let $Q : X^* \times [0,T] \to \mathbb{R}$ be a sensitivity-1 quality function, and let $S \in X^*$ be a database s.t. $Q(S,\cdot)$ is concave. Let $\alpha \leq \frac{1}{2}$ and let $\beta, \epsilon, \delta, r$ be s.t. $L(S,0) \triangleq \max_i\{Q(S,i)\} \geq r \geq \left(\frac{2}{\alpha}\right)^{\log^*(T)} \frac{16}{\alpha\epsilon} \ln\left(\frac{32}{\beta\delta}\right)$. When executed on $S, [0,T], r, \alpha, \epsilon, \delta$, algorithm Rec fails to outputs an index j s.t. $Q(S,j) \geq (1-\alpha)r$ with probability at most $2\beta\log^*(T)$.*

Remark 2. The computational efficiency of algorithm *Rec* depends on the quality function $Q(\cdot,\cdot)$. Note, however, that it suffices to efficiently implement the top level call (i.e., without the recursion). This is because an iteration of algorithm *Rec*, operating on a range $[0,T]$, can easily be implemented in time poly(T), and the range given as input to recursive calls is logarithmic in the size of the initial range.

Algorithm *Rec* can be used as a proper $(\alpha,\beta,\epsilon,\delta,m)$-private learner for $\texttt{INTERVAL}_d$. The details of this straight forward application are omitted from this extended abstract.

Theorem 2. *There exists an efficient proper $(\alpha,\beta,\epsilon,\delta,m)$-PPAC learner for $\texttt{INTERVAL}_d$, where $m = O_{\alpha,\beta,\epsilon}\left(16^{\log^*(d)}\sqrt{\log^*(d)\log(\frac{1}{\delta})}\log\left(\frac{1}{\delta}\log^*(d)\right)\right)$.*

4 Sanitization with Approximate Privacy

Beimel et al. [1] showed that every pure ϵ-private sanitizer for $\texttt{POINT}_d$, must operate on databases of $\Omega(d)$ elements. In this section we state the existence of an

Algorithm *Rec*
Inputs: range $[0,T]$, quality function Q, quality promise r, parameters α, ϵ, δ, and a sample S.

1. If $T \leq 32$, then use the exponential mechanism with the quality function Q and the parameter ϵ to choose and return an index $j \in [0,\ldots,T]$.
2. Let T' be the smallest power of 2 s.t. $T' \geq T$, and define $Q(S,i) = 0$ for $T < i \leq T'$.
3. For $0 \leq j \leq \log(T')$ let $L(S,j) = \max\limits_{\substack{[a,b]\subseteq[0,T'] \\ b-a+1=2^j}} \left(\min\limits_{i\in[a,b]} \Big(Q(S,i)\Big) \right)$. For $j = \log(T')+1$ let $L(S,j) = \min\{0, L(S,\log(T'))\}$.
   % If $L(S,j) = x$ then in every interval $I \subseteq [0,T']$ of length 2^j there exists a point $i \in I$ s.t. $Q(S,i) \leq x$. Moreover, there exists an interval $I \subseteq [0,T']$ of length 2^j s.t. $Q(S,i) \geq x$ for all $i \in I$. Note that, as $L(S,j+1)$ maximizes the minimum over intervals bigger than I, it must be bounded by x.
4. Define the function $q(S,j) = \min\Big(L(S,j) - (1-\alpha)r, r - L(S,j+1)\Big)$ where $0 \leq j \leq \log(T')$.
   % If $q(S,j)$ is high for some j, then there exists an interval $I = [a, a+2^j-1]$ s.t. every $i \in I$ has a quality $Q(S,i) >> (1-\alpha)r$, and for every interval $I' = [a', a'+2^{j+1}-1]$ there exists $i' \in I'$ with quality $Q(S,i) << r$.
5. Let $R = \frac{\alpha}{2}r$.
   % R is the promise parameter for the recursive call. Note that for the maximal j with $L(S,j) \geq (1-\frac{\alpha}{2})r$ we get $q(S,j) \geq \frac{\alpha}{2}r = \mathrm{R}$.
6. Execute *Rec* recursively on the range $\{0,\ldots,\log(T')\}$, the quality function $q(\cdot,\cdot)$, the promise R, and α, ϵ, δ. Denote the returned value by k, and let $K = 2^k$.
   % Assuming the recursive call was successful, k is s.t. $q(S,k) \geq (1-\alpha)R = (1-\alpha)\frac{\alpha}{2}r$. That is, $L(S,k) \geq (1-\frac{\alpha}{2}-\frac{\alpha^2}{2})r$ and $L(S,k+1) \leq (1-\frac{\alpha}{2}+\frac{\alpha^2}{2})r$.
7. Divide $[0,T']$ into the following intervals of length $8K$ (the last ones might be trimmed):
   $A_1 = [0, 8K-1],\ A_2 = [8K, 16K-1],\ A_3 = [16K, 24K-1], \ldots$
   $B_1 = [4K, 12K-1],\ B_2 = [12K, 20K-1],\ B_3 = [20K, 28K-1], \ldots$
   % We show that in at least one of those two partitions (say the $\{A_i\}$'s), there exists a "good" interval A_g s.t. $Q(S,i) = r$ for some $i \in A_g$, and $Q(S,i) \leq (1-\frac{\alpha}{2}+\frac{\alpha^2}{2})r$ for all $i \in \{0,\ldots,T\} \setminus A_g$.
8. For every such interval $I \in \{A_i\} \cup \{B_i\}$ let $u(S,I) = \max\limits_{i\in I} \Big(Q(S,i)\Big)$.
9. Use algorithm $\mathcal{A}_{dist}$ with parameters ϵ, δ and the quality function $u(\cdot,\cdot)$, once to choose an interval $A \in \{A_i\}$, and once more to choose an interval $B \in \{B_i\}$.
   % By the properties of $\mathcal{A}_{dist}$, w.h.p. at least one of the returned A and B is "good".
10. Denote $A = [a,b]$ and $B = [c,d]$, and define $H = \{a + i\frac{K}{2} : 0 \leq i \leq 15\} \cup \{b + i\frac{K}{2} : 0 \leq i \leq 15\}$.
    % We show that H contains an index with high quality.
11. Use the exponential mechanism with the quality function $Q(\cdot,\cdot)$ and parameter ϵ to choose and return an index $j \in H$.

Fig. 1. Algorithm *Rec*

(ϵ, δ)-private sanitizer for $\texttt{POINT}_d$ with sample complexity $O_{\alpha,\beta,\epsilon,\delta}(1)$. This separates the database size necessary for $(\epsilon, 0)$-private sanitizers from the database size sufficient for (ϵ, δ)-private sanitizers.

Recall that in our private PAC learner for $\texttt{POINT}_d$, given a typical labeled sample, there exists a unique concept in the class that stands out (we used algorithm $\mathcal{A}_{dist}$ to identify it). This is not the case in the context of sanitization, as a given database S may have many α-close sanitized databases $\hat{S}$. We will overcome this issue using the following private tool for approximating a restricted class of choosing problems.

4.1 The Choosing Mechanism

A function $q : X^* \times \mathcal{F} \to \mathbb{N}$ defines an *optimization problem* over the domain X and solution set $\mathcal{F}$: Given a dataset S over domain X choose $f \in \mathcal{F}$ which (approximately) maximizes $q(S, f)$. We are interested in a subset of these optimization problems, which we call *bounded-growth choice problems*. For this section we think of a database $S \subseteq X^*$ as a multiset.

Definition 11. *Given q and S define $\mathrm{opt}_q(S) = \max_{f\in\mathcal{F}}\{q(S, f)\}$. A solution $f \in \mathcal{F}$ is called* α-good *for a database S if $q(S, f) \geq \mathrm{opt}_q(S) - \alpha|S|$.*

Definition 12. *A scoring function $q : X^* \times \mathcal{F} \to \mathbb{N}$ is* k-bounded-growth *if:*

1. *$q(\emptyset, f) = 0$ for all $f \in \mathcal{F}$.*
2. *If $S_2 = S_1 \cup \{x\}$, then (i) $q(S_2, f) \geq q(S_1, f) \geq q(S_2, f) - 1$ for all $f \in \mathcal{F}$; and (ii) there are at most k solutions $f \in \mathcal{F}$ s.t. $q(S_1, f) < q(S_2, f)$.*

In words, the second requirement means that (i) Adding an element to the database could either have no effect on the score of a solution f, or can increase the score by exactly 1; and (ii) There could be at most k solutions whose scores are increased (by 1). Note that a k-bounded-growth scoring function is, in particular, a sensitivity-1 function as two neighboring S_1, S_2 must be of the form $D \cup \{x_1\}$ and $D \cup \{x_2\}$ respectively. Hence, $q(S_1, f) - q(S_2, f) \leq q(D, f) + 1 - q(D, f) = 1$ for every solution f.

The choosing mechanism below is a private algorithm for approximately solving bounded-growth choice problems. Step 1 of the algorithm checks whether a good solutions exist, as otherwise any solution is approximately optimal (and the mechanism returns $\perp$). Step 2 invokes the exponential mechanism, but with the *small* set $G(S)$ instead of $\mathcal{F}$. We get the following lemma.

Choosing Mechanism

Input: a database S of $m \geq \frac{16}{\alpha\epsilon} \cdot \max\left((\frac{\epsilon}{4} + \ln(\frac{1}{\delta})), \ln(\frac{16k}{\alpha\beta\epsilon})\right)$ elements, and parameters $\alpha, \beta, \epsilon, \delta$.

1. Set $\mathrm{best}(S) = \max_{f\in\mathcal{F}}\{q(S, f)\} + \mathrm{Lap}(\frac{4}{\epsilon})$. If $\mathrm{best}(S) < \frac{\alpha m}{2}$ then halt and return $\perp$.
2. Let $G(S) = \{f \in \mathcal{F} : q(S, f) \geq 1\}$. Choose and return $f \in G(S)$ using the exponential mechanism with parameter $\frac{\epsilon}{2}$.

Lemma 4. *When q is a k-bounded-growth quality function, the choosing mechanism is (ϵ, δ)-differentially private. Moreover, given a database S the choosing mechanism outputs an α-good solution for S with probability at least $1 - \beta$.*

4.2 (ϵ, δ)-Private Sanitiazers

Using the above choosing mechanism, we construct a private sanitizer for $\texttt{POINT}_d$. We also construct a recursive sanitizer for $\texttt{INTERVAL}_d$, using both algorithm *Rec* and the choosing mechanism. Here we only state the results.

Theorem 3. *Fix $\alpha, \beta, \epsilon, \delta$. There exists an efficient $(\alpha, \beta, \epsilon, \delta, m)$-improper- sanitizer for $\texttt{POINT}_d$, where $m = O\left(\frac{1}{\alpha^{1.5}\epsilon}\sqrt{\ln(\frac{1}{\delta})}\ln(\frac{1}{\alpha\beta\epsilon\delta})\right)$.*

Theorem 4. *Fix $\alpha, \beta, \epsilon, \delta$. There exists an efficient $(\alpha, \beta, \epsilon, \delta, m)$-proper- sanitizer for $\texttt{INTERVAL}_d$, where $m = O\left(\frac{1}{\alpha^{2.5}\epsilon}8^{\log^*(d)}\sqrt{\log^*(d)}\log(\frac{1}{\alpha\delta})\log\left(\frac{\log^*(d)}{\alpha\beta\epsilon\delta}\right)\right)$.*

5 Sanitization and Proper Private PAC

Similar techniques are used for both data sanitization and private learning, suggesting relationships between the two. We now explore one such relationship in proving a lower bound on the sample complexity needed for sanitization (under pure differential privacy). In particular, we show a *reduction* from the task of private learning to the task of data sanitization, and then use a lower bound on private learners to derive a lower bound on data sanitization.

Notation. We will refer an element of X_{d+1} as $\boldsymbol{x} \circ y$, where $\boldsymbol{x} \in X_d$, and $y \in \{0, 1\}$.

5.1 Sanitization Implies Proper PPAC

For a given predicate c over X_d, we define the predicate c^{label} over X_{d+1} as

$$c^{label}(\boldsymbol{x} \circ y) = \begin{cases} 1, & c(\boldsymbol{x}) \neq y. \\ 0, & c(\boldsymbol{x}) = y. \end{cases}$$

Note that $c^{label}(\boldsymbol{x} \circ \sigma) = \sigma \oplus c(\boldsymbol{x})$ for $\sigma \in \{0, 1\}$. For a given class of predicates C over X_d, we define $C^{label} = \{c^{label} \ : \ c \in C\}$.

The next theorem states that for every concept class C, a sanitizer for C^{label} implies a private learner for C.

Theorem 5. *Let $\alpha, \epsilon \leq \frac{1}{8}$, and let C be a class of predicates. If there exists an $(\alpha, \beta, \epsilon, m)$-sanitizer A for C^{label}, then there exists a proper $((3\alpha + 2\beta), 2\beta, \epsilon, t)$-PPAC learner for C, where $t = \mathcal{O}_{\alpha,\beta}(m)$.*

Remark 3. Given an efficient proper-sanitizer for a class C, and assuming the existence of an efficient *non-private* learner for C, this reduction results in an efficient *private* learner for the class C.

Using the above reduction together with known lower bounds on the sample complexity of private learners, we get:

Theorem 6. *There exists an explicit concept class C over X_d such that every $(\alpha, \beta, \epsilon, m)$-sanitizer for C requires databases of size*

$$m = \Omega\left(\frac{1}{\alpha\epsilon}\,\mathrm{VC}(C) \cdot \log|X_d|\right).$$

6 Generic Label-Private Learner

The model of *label privacy* was defined as a relaxation of private learning, where privacy must only be preserved for the *labels* of the elements in the database, and not necessarily for the elements themselves. This is a reasonable privacy requirement when the elements are public and only their labels are private.

Consider a database $S = (x_i, y_i)_{i=1}^m$ containing labeled points from some domain X. We denote $S_x = (x_i)_{i=1}^m \in X^m$, and $S_y = (y_i)_{i=1}^m \in \{0,1\}^m$.

Definition 13 (Label-Private Learner [5]). *Let A be an algorithm that gets as input a database $S_x \in X^m$ and its labeling $S_y \in \{0,1\}^m$. Algorithm A is an* $(\alpha, \beta, \epsilon, m)$-Label Private PAC Learner *for a concept class $\mathcal{C}$ over X if*

Privacy. *$\forall S_x \in X^m$, algorithm $A(S_x, \cdot) = A_{S_x}(\cdot)$ is ϵ-differentially private (as in Definition 1);*
Utility. *Algorithm A is an* (α, β, m)-PAC learner *for $\mathcal{C}$ (as in Definition 3).*

Chaudhuri et al. [5] show lower bounds on the sample complexity of label-private learners for a class C in terms of its doubling dimension. As the next theorem states, the correct measure for characterizing the sample complexity of such learners is the VC dimension, and the sample complexity of label-private learners is actually of the same order as that of non-private learners (assuming α, β and ϵ are constants).

Theorem 7. *Let C be a concept class over a domain X. For every α, β, ϵ, there exists an $(\alpha, \beta, \epsilon, m)$-Label Private PAC learner for C, where $m = O_{\alpha,\beta,\epsilon}(VC(C))$. The learner might not be efficient.*

Acknowledgments. We thank Salil Vadhan and Jon Ullman for helpful discussions of ideas in this work.

References

1. Beimel, A., Kasiviswanathan, S.P., Nissim, K.: Bounds on the sample complexity for private learning and private data release. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 437–454. Springer, Heidelberg (2010)
2. Beimel, A., Nissim, K., Stemmer, U.: Characterizing the sample complexity of private learners. In: ITCS, pp. 97–110 (2013)

3. Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: The SuLQ framework. In: PODS, pp. 128–138. ACM (2005)
4. Blum, A., Ligett, K., Roth, A.: A learning theory approach to noninteractive database privacy. J. ACM 60(2), 12:1–12:25 (2013)
5. Chaudhuri, K., Hsu, D.: Sample complexity bounds for differentially private learning. In: COLT, vol. 19, pp. 155–186 (2011)
6. De, A.: Lower bounds in differential privacy. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 321–338. Springer, Heidelberg (2012)
7. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: Privacy via distributed noise generation. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 486–503. Springer, Heidelberg (2006)
8. Dwork, C., Lei, J.: Differential privacy and robust statistics. In: STOC 2009, pp. 371–380. ACM, New York (2009)
9. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
10. Dwork, C., Naor, M., Reingold, O., Rothblum, G., Vadhan, S.: On the complexity of differentially private data release. In: STOC, pp. 381–390. ACM (2009)
11. Gupta, A., Hardt, M., Roth, A., Ullman, J.: Privately releasing conjunctions and the statistical query barrier. In: STOC, pp. 803–812. ACM, New York (2011)
12. Hardt, M., Talwar, K.: On the geometry of differential privacy. In: STOC,7 pp. 705–714 (2010)
13. Hardt, M.A.W.: A Study of Privacy and Fairness in Sensitive Data Analysis. PhD thesis, Princeton University (2011)
14. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately? In: FOCS, pp. 531–540. IEEE Computer Society (2008)
15. Kearns, M.J.: Efficient noise-tolerant learning from statistical queries. Journal of the ACM 45(6), 983–1006 (1998); Preliminary version in Proceedings of STOC 1993
16. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: FOCS, pp. 94–103. IEEE (2007)
17. Roth, A.: Differential privacy and the fat-shattering dimension of linear queries. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, pp. 683–695. Springer, Heidelberg (2010)
18. Smith, A., Thakurta, A.: Differentially private feature selection via stability arguments, and the robustness of the lasso. Manuscript (2012)
19. Ullman, J.: Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. CoRR, abs/1207.6945 (2012)
20. Ullman, J., Vadhan, S.: PCPs and the hardness of generating private synthetic data. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 400–416. Springer, Heidelberg (2011)
21. Valiant, L.G.: A theory of the learnable. Communications of the ACM 27, 1134–1142 (1984)

Phase Coexistence and Slow Mixing for the Hard-Core Model on $\mathbb{Z}^2$

Antonio Blanca[1,⋆], David Galvin[2,⋆⋆], Dana Randall[3,⋆⋆⋆], and Prasad Tetali[4,†]

[1] Computer Science Division, U.C. Berkeley, Berkeley, CA 94720
ablanca@cs.berkeley.edu
[2] Department of Mathematics, University of Notre Dame, Notre Dame, IN 46656
dgalvin1@nd.edu
[3] School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332-0765
randall@cc.gatech.edu
[4] School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332-0280
tetali@math.gatech.edu

Abstract. For the hard-core model (independent sets) on $\mathbb{Z}^2$ with fugacity λ, we give the first explicit result for phase coexistence by showing that there are multiple Gibbs states for all $\lambda > 5.3646$. Our proof begins along the lines of the standard Peierls argument, but we add two significant innovations. First, building on the idea of fault lines introduced by Randall [19], we construct an event that distinguishes two boundary conditions and yet always has long contours associated with it, obviating the need to accurately enumerate short contours. Second, we obtain vastly improved bounds on the number of contours by relating them to a new class of self-avoiding walks on an oriented version of $\mathbb{Z}^2$. We also extend our characterization of fault lines to show that local Markov chains will mix slowly when $\lambda > 5.3646$ on lattice regions with periodic (toroidal) boundary conditions and when $\lambda > 7.1031$ with non-periodic (free) boundary conditions. The arguments here rely on a careful analysis that relates contours to taxi walks and represent a sevenfold improvement to the previously best known values of λ [19].

1 Introduction

The hard-core model was introduced in statistical physics as a model for lattice gases, where each molecule occupies non-trivial space in the lattice, requiring occupied sites to be non-adjacent. Viewing a lattice such as $\mathbb{Z}^d$ as a graph, allowed configurations of molecules naturally correspond to independent sets in the graph.

Given a finite graph G, let Ω be the set of independent sets of G. Given a (fixed) *activity* (or *fugacity*) $\lambda \in \mathbb{R}^+$, the weight associated with each independent set I is $w(I) = \lambda^{|I|}$. The associated *Gibbs* (or *Boltzmann*) *distribution* $\mu = \mu_{G,\lambda}$ is defined on Ω, assuming G is finite, as $\mu(I) = w(I)/Z$, where the normalizing constant $Z = Z(G,\lambda) = \sum_{J\in\Omega} w(J)$ is commonly called the *partition function*. Physicists are interested in the behavior of models on an infinite graph (such as the integer lattice

⋆ Supported by a UC Berkeley Chancellor's Fellowship for Graduate Study and an NSF Graduate Research Fellowship.
⋆⋆ Supported in part by the Simons Foundation and by NSA grant number H98230-13-1-0248.
⋆⋆⋆ Supported in part by NSF grant CCF-1219202.
† Supported in part by NSF grant DMS-1101447.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 379–394, 2013.

$\mathbb{Z}^d$), where the Gibbs measure is defined as a certain weak limit with appropriate conditional probabilities. For many models it is believed that as a parameter of the system is varied – such as the inverse temperature β for the Ising model or the activity λ for the hard-core model – the system undergoes a phase transition at a critical point.

For the classical Ising model, Onsager, in seminal work [17], established the precise value of the critical temperature $\beta_c(\mathbb{Z}^2)$ to be $\log(1+\sqrt{2})$. Only recently have the analogous values for the (more general) q-state Potts model been established [3]. Establishing such a precise value for the hard-core model with currently available methods seems nearly impossible. Even the *existence* of such a (unique) critical activity λ_c, where there is a transition from a unique Gibbs state to the coexistence of multiple Gibbs states, remains conjectural for $\mathbb{Z}^d$ ($d \geq 2$; it is folklore that there is no such transition for $d = 1$), while it is simply untrue for general graphs (even general trees, in fact, thanks to a result of Brightwell et al. [7]). Regardless, a non-rigorous prediction from the statistical physics literature [2] suggests $\lambda_c \approx 3.796$ for $\mathbb{Z}^2$.

Thus, from a statistical physics or probability viewpoint, understanding the precise dependence on λ for the existence of unique or multiple Gibbs states is a natural and challenging problem. Moreover, breakthrough works of Weitz [25] and Sly [22] have recently identified $\lambda_c(\mathbb{T}_\Delta)$ – the critical activity for the hard-core model on an infinite Δ-regular tree – as a *computational* threshold where estimating the hard-core partition function on general Δ-regular graphs undergoes a transition beyond which there is no PTAS unless $NP = RP$, further motivating the study of physical transitions and their computational implications. While it is not surprising that for many problems computing the partition function *exactly* is intractable, it is remarkable that even approximating it for the hard-core model above a certain critical threshold also turns out to be hard.

Starting with Dobrushin [8] in 1968, physicists have been developing techniques to characterize the regimes on either side of λ_c for the hard-core model. Most attention has focused on establishing ever larger values of λ below which there is always uniqueness of phase. The problem has proved to be a fruitful one for the blending of ideas from physics, discrete probability and theoretical computer science (see, e.g., [18] [4], [25]). The state of the art is work by Vera et al. [24], expanding on ideas from Weitz [25] and Restrepo et al. [20], establishing uniqueness for $\lambda < 2.48$.

Much less is known about the regime of phase coexistence. Dobrushin [8] established phase coexistence for all $\lambda > C$, but did not explicitly calculate C. Borgs had estimated that $C = 80$ was the *theoretical* lower limit of Dobrushin's argument [5], but a recent computation by the second author suggests that the *actual* consequence of the argument is more like $C \approx 300$. We are now prepared to state our first main result.

Theorem 1. *For $\lambda > 5.3646$, the hard-core model on $\mathbb{Z}^2$ with activity λ admits multiple Gibbs states.*

From a computational standpoint, there are two natural questions to ask concerning the hard-core model on a finite graph. Can the partition function be approximated, and how easy is it to sample from a given Gibbs distribution? For both, a powerful method is given by Markov chain algorithms – carefully constructed random walks on the space of independent sets of a graph whose equilibrium distributions are the desired Gibbs distributions. One of the most commonly studied families are the *local-update* chains, such as Glauber dynamics, that change a bounded number of vertices at each step.

The efficiency of the Markov chain method relies on the underlying chain being rapidly mixing; that is, it must fairly quickly reach a distribution that is close to stationary. For many problems, local chains seem to mix rapidly below some critical point, while mixing slowly above that point. Most notably for the Ising model on $\mathbb{Z}^2$, simple local Markov chains are rapidly mixing (in fact, with optimal rate) for $\beta < \beta_c(\mathbb{Z}^2)$ and slowly mixing for $\beta > \beta_c(\mathbb{Z}^2)$. Recently the Ising picture was completed by Lubezky and Sly [13], who showed polynomial mixing at $\beta = \beta_c(\mathbb{Z}^2)$.

Once again, the known bounds are less sharp for the hard-core model. Luby and Vigoda [14] showed that Glauber dynamics on independent sets is fast when $\lambda \leq 1$ on the 2-dimensional lattice and torus. Weitz [25] reduced the analysis on the grid to the tree, thus establishing that in this same setting Glauber dynamics is fast up to the critical point for the 4-regular tree, in effect for $\lambda < 1.6875$. Again, the best result to date is due to Vera et al. [24] who proved that Glauber dynamics on the space of hard-core configurations on boxes in $\mathbb{Z}^2$ is rapidly mixing for all $\lambda < 2.48$.

As with phase-coexistence, it is believed there is a critical value λ_c^{mix} at which Glauber dynamics for sampling hard-core configurations flips from mixing in time polynomial in n, to exponential in n, and that it coincides with λ_c. Borgs et al. [6] showed that Glauber dynamics is slow on toroidal lattice regions in $\mathbb{Z}^d$ (for $d \geq 2$), when λ is sufficiently large, in particular establishing a finite constant above which mixing is slow on $\mathbb{Z}^2$. The first effective bound was provided by Randall [19], who showed slow mixing for $\lambda > 50.526$ on boxes with periodical boundary conditions, and for $\lambda > 56.812$ on boxes with free boundary [1] (but did not address the question of phase coexistence).

Our second main result establishes slow mixing of Glauber dynamics on boxes in $\mathbb{Z}^2$ for λ that is an order of magnitude lower than the previously best known bounds.

Theorem 2. *For $\lambda > 5.3646$, the mixing time of Glauber dynamics for the hard-core model on n by n boxes in $\mathbb{Z}^2$ with periodic boundary conditions and activity λ is exponential in n. For free boundary conditions, we have the same result with $\lambda > 7.1031$.*

The proofs of Theorems 1 and 2 utilize combinatorial, computational and physical insights. The standard approach to showing multiple Gibbs distributions, introduced by Dobrushin [8], is to consider the limiting distributions corresponding to two different boundary conditions on boxes in the lattice centered at the origin, and find a statistic that separates these two limits. For the hard-core model, it suffices (see [4]) to compare the *even boundary condition* – all vertices on the boundary of a box at an even distance from the origin are occupied – and its counterpart the odd boundary condition, and the distinguishing statistic is typically the occupation of the origin. Under odd boundary condition the origin should be unlikely to be occupied, since independent sets with odd boundary and (even) origin occupied must have a *contour* – a two-layer thick unoccupied loop of vertices separating an inner region around the origin that is in "even phase" from an outer region near the boundary that is in "odd phase". For large enough λ, such an unoccupied layer is costly, and so such configurations are unlikely.

As we will see presently, the effectiveness of this approach, known as a *Peierls argument*, is driven by the number of contours of each possible length – better upper bounds

[1] Note that stronger bounds were reported in [19] due to a missing factor of 2 in the computations; see `http://www.math.gatech.edu/~randall/ind-fix.pdf` for the corrected version.

on the number of contours translate directly to better upper bounds on λ_c. Previous (unpublished) work on phase coexistence in the hard-core model on $\mathbb{Z}^2$ had viewed contours as simple polygons in $\mathbb{Z}^2$, which are closely related to the very well studied family of self-avoiding walks. While this is essentially the best possible point of view when applying the Peierls argument on the Ising model, it is far from optimal for the hard-core model. One of the two major breakthroughs of the present paper is the discovery that hard-core contours, if appropriately defined, can be viewed as simple polygons in the oriented *Manhattan lattice* (orient edges of $\mathbb{Z}^2$ that are parallel to the x-axis (resp. y-axis) positively if their y-coordinate (resp. x-coordinate) is even, and negatively otherwise), with the additional constraint that contours cannot make two consecutive turns. The number of such polygons can be understood by analyzing a new class of self-avoiding walks, that we refer to as *taxi walks*. The number of taxi walks turns out to be significantly smaller than the number of ordinary self-avoiding walks, leading to much better bounds on λ_c than could possibly have been obtained previously.

The number $\widetilde{c}_n$ of taxi walks of length n is asymptotically controlled by a single number $\mu_t > 0$, the *taxi walk connective constant*, in the sense that $(\widetilde{c}_n)^{1/n} \sim \mu_t$ as $n \to \infty$. Adapting methods of Alm [1] we obtain good estimates on μ_t, allowing us to understand $\widetilde{c}_n$ for large n. It is difficult to control $\widetilde{c}_n$ for small n, however, presenting a major stumbling block to the effectiveness of the Peierls argument. Using the statistic "occupation of origin" to distinguish the two boundary conditions, one inevitably has to control $\widetilde{c}_n$ for both small and large n. The lack of precise information about the number of short contours leads to discrepancies between theoretical lower limits and actual bounds, such as that between $C = 80$ (theoretical best possible) and $C \approx 300$ for phase coexistence on $\mathbb{Z}^2$, discussed earlier.

The second breakthrough of the present paper is the idea of using an event to distinguish the two boundary conditions with the property that every independent set in the event has a *long* contour. This allows us to focus exclusively on the asymptotic growth rate of contours/taxi walks, and obviates the need for an analysis of short contours. The immediate result of this breakthrough is that the actual limits of our arguments agree exactly with their theoretical counterparts. The distinguishing event we use extends the idea of *fault lines*, which we discuss in more detail below in the context of slow mixing.

The traditional argument for slow mixing is based on the observation that when λ is large, the Gibbs distribution favors dense configurations, and Glauber dynamics will take exponential time to converge to equilibrium. The slow convergence arises because the Gibbs distribution is bimodal: dense configurations lie predominantly on either the odd or the even sublattice, while configurations that are roughly half odd and half even have much smaller probability. Since Glauber dynamics changes the numbers of even and odd vertices by at most 1 in each step, the Markov chain has a bottleneck.

Our work builds on a novel idea from [19], namely using *fault lines* to establish slow mixing for Glauber dynamics on hard-core configurations for large λ, improving upon what was best known at that time. Randall [19] gave an improvement by realizing that the state space could be partitioned according to certain *topological obstructions* in configurations, rather than the relative numbers of odd or even vertices. This approach gives better bounds on λ, and also greatly simplifies the calculations. First consider an $n \times n$ lattice region G with free (non-periodic) boundary conditions. A configuration I

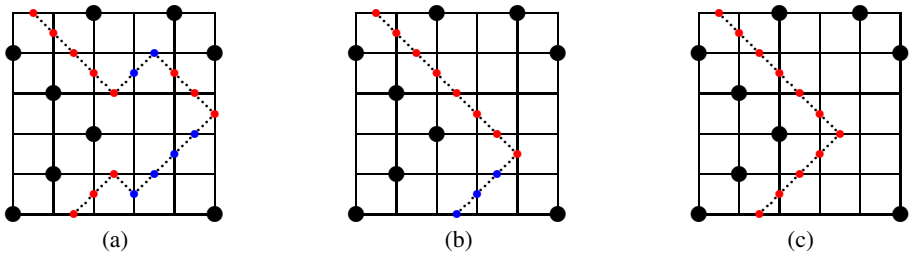


Fig. 1. Independent sets with (a) a spanning path with four alternation points in $G_\diamond$, (b) a fault line with one alternation point and (c) a fault line with no alternation points

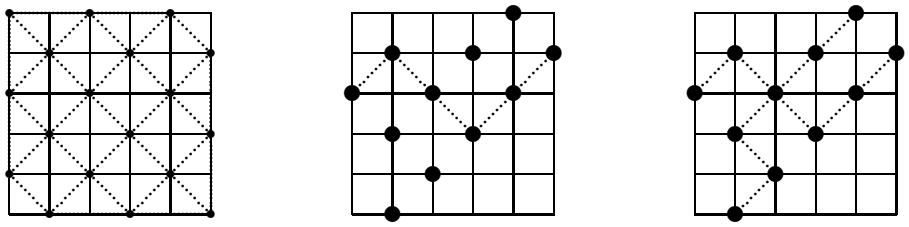
is said to have a *fault line* if there is a width two path of unoccupied vertices in I from the top of G to the bottom or from the left boundary of G to the right. Configurations without a fault line must have a *cross* of occupied vertices in either the even or the odd sublattices forming a connected path in G^2 from both the top to the bottom and from the left to the right of G, where G^2 connects vertices at distance 2 in G. Roughly speaking the set of configurations that have a fault line forms a cut set that must be crossed to move from a configuration that has an odd cross to one with an even cross, and it was shown that fault lines are exponentially unlikely when λ is large. Likewise, if $\widehat{G}$ is an $n \times n$ region with periodic boundary conditions, it was shown that either there is an odd or an even cross forming non-contractible loops in two different directions or there is a pair of non-contractible fault lines, allowing for a similar argument.

We improve the argument by refining our consideration of fault lines, which had previously been characterized as (rotated) self-avoiding walks in $\mathbb{Z}^2$. Here we observe that, suitably modified, they are in fact taxi walks and so the machinery developed for phase coexistence can be brought to bear in the mixing context.

2 Combinatorial Background: Crosses, Fault Lines and Taxi Walks

We begin by introducing the notions of crosses and fault lines. Let $G = (V, E)$ be a simply connected region in $\mathbb{Z}^2$, say the $n \times n$ square. We define the graph $G_\diamond = (V_\diamond, E_\diamond)$ as follows. The vertices $V_\diamond$ are the midpoints of edges in E. Vertices u and v in $V_\diamond$ are connected by an edge in $E_\diamond$ if and only if they are the midpoints of incident edges in E that are perpendicular. Notice that $G_\diamond$ is a region in a smaller Cartesian lattice that has been rotated by 45 degrees. We will also make use of the *even and odd subgraphs* of G. For $b \in \{0, 1\}$, let $G_b = (V_b, E_b)$ be the graph whose vertex set $V_b \subseteq V$ contains all vertices with parity b (i.e., the sum of their coordinates has parity b), with $(u, v) \in E_b$ if u and v are at Hamming distance 2. We refer to G_0 and G_1 as the even and odd subgraphs. The graphs $G_\diamond, G_0$ and G_1 play a central role in defining features of independent sets that determine distinguishing events in our proof of phase coexistence and the partition of the state space for our proofs of slow mixing.

Given an independent set $I \in \Omega$, we say that a simple path p in $G_\diamond$ is *spanning* if it extends from the top boundary of $G_\diamond$ to the bottom, or from the left boundary to the



(a) (b) (c)

Fig. 2. The figure shows: (a) G_1, (b) an independent set with an odd bridge in G_1, and (c) two odd bridges forming an odd cross in G_1

right, and each vertex in p corresponds to an edge in G such that both endpoints are unoccupied in I. It will be convenient to color the vertices on p according to whether the corresponding edge in G has an odd or an even vertex to its left as we traverse the path. Specifically, suppose vertex $v \in V_\Diamond$ on the path p bisects an edge $e_v \in E$. Color v *blue* if the odd vertex in e_v is to the left when the path crosses v, and *red* otherwise (note that each $v \in E$ has one odd and one even endvertex). When the color of the vertices along the path changes, we have an *alternation point* (see Fig. 1 (a) and (b)). It was shown in [19] that if an independent set has a spanning path, then it must also have one with zero or one alternation points. We call such a path a *fault line*, and let $\Omega_{\mathcal{F}}$ be the set of independent sets in Ω with at least one fault line (see Fig. 1 (b) and (c)).

We say that $I \in \Omega$ has an *even bridge* if there is a path from the left to the right boundary or from the top to the bottom boundary in G_0 consisting of occupied vertices in I. Similarly, we say it has an *odd bridge* if it traverses G_1 in either direction. We say that I has a *cross* if it has both left-right and a top-bottom bridges. See Fig. 2.

Notice that if an independent set has an even top-bottom bridge it cannot have an odd left-right bridge, so if it has a cross, both of its bridges must have the same parity. We let $\Omega_0 \subseteq \Omega$ be the set of configurations that contain an even cross and let $\Omega_1 \subseteq \Omega$ be the set of those with an odd cross.

We can now partition the state space Ω into three sets, with one separating the other two; this partition is critical to the proofs of both Theorem 1 and Theorem 2. The following lemma was proven in [19].

Lemma 1. *The set of independent sets on G can be partitioned into sets $\Omega_{\mathcal{F}}$, Ω_0 and Ω_1, consisting of configurations with a fault line, an even cross or an odd cross. If $I \in \Omega_0$ and $I' \in \Omega_1$ then $|I \triangle I'| > 1$.*

It will be useful to extend these definitions to the torus as well. Let n be even, and let $\widehat{G}$ be the $n \times n$ toroidal region $\{0, \ldots, n-1\} \times \{0, \ldots n-1\}$, where $v = (v_1, v_2)$ and $u = (u_1, u_2)$ are connected if $v_1 = u_1 \pm 1 (\text{mod } n)$ and $v_2 = u_2$ or $v_2 = u_2 \pm 1 (\text{mod } n)$ and $v_1 = u_1$. Let $\widehat{\Omega}$ be the set of independent sets on $\widehat{G}$ and let $\widehat{\pi}$ be the Gibbs distribution. As before, we consider Glauber dynamics that connect configurations with symmetric difference of size one. We define $\widehat{G}_\Diamond$, $\widehat{G}_0$ and $\widehat{G}_1$ as above to represent the graph connecting the midpoints of perpendicular edges (including the boundary edges), and the odd and even subgraphs. As with $\widehat{G}$, all of these have toroidal boundary conditions.

Given $I \in \widehat{\Omega}$, we say that I has a *fault* $F = (F_1, F_2)$ if there are a *pair* of vertex-disjoint non-contractible cycles F_1, F_2 in $\widehat{G}_\diamond$ whose vertices correspond to edges in $\widehat{G}$ whose endpoints are unoccupied, and such that the vertices on each cycle are all red or all blue (i.e., the endpoints in $\widehat{G}$ to one side of either cycle have the same parity). We say that I has a *cross* if it has at least two non-contractible cycles of occupied sites in I with different winding numbers. The next lemma (from [19]) utilizes faults to partition $\widehat{\Omega}$.

Lemma 2. *The set of independent sets on $\widehat{G}$ can be partitioned into sets $\widehat{\Omega}_{\mathcal{F}}, \widehat{\Omega}_0, \widehat{\Omega}_1$, consisting of configurations with a fault, an even cross or an odd cross. If $I \in \widehat{\Omega}_0$ and $I' \in \widehat{\Omega}_1$ then $|I \triangle I'| > 1$.*

The strategy for the proofs of phase coexistence and slow mixing will be to use a Peierls argument to define a map from $\Omega_{\mathcal{F}}$ to Ω that takes configurations with fault lines to ones with exponentially larger weight. The map is not injective, however, so we need to be careful about how large the pre-image of a configuration can be, and for this it is necessary to get a good bound on the number of fault lines. In [19] the number of fault lines was bounded by the number of self-avoiding walks in $G_\diamond$ (or $\widehat{G}_\diamond$ on the torus). However, this is a gross overcount because this includes all spanning paths with an arbitrary number of alternation points. Instead, we can get much better bounds by only counting self-avoiding walks with zero or one alternation points.

To begin formalizing this idea, we put an orientation on the edges of $G_\diamond$. Each edge $(u, v) \in E_\diamond$ corresponds to two edges in E that share a vertex $w \in V$. We orient the edge "clockwise" around w if w is even and "counterclockwise" if w is odd. For paths with zero alternation points, all of the edges must be oriented in the same direction (with respect to this edge orientation). If we rotate $G_\diamond$ so that the edges are axis aligned, then this simply means that the horizontal (resp. vertical) edges alternate direction according to the parity of the y- (resp. x-) coordinates, like in many well-known metropolises.

We now define taxi walks. Let $\mathbb{Z}^2$ be an orientation of $\mathbb{Z}^2$ in which an edge parallel to the x-axis (resp. y-axis) is oriented in the positive x-direction if its y-coordinate is even (resp. oriented in the positive y-direction if its x-coordinate is even), and is oriented in the negative direction otherwise (note that this agrees with the orientation placed on $G_\diamond$ above). It is common to refer to $\mathbb{Z}^2$ as the *Manhattan lattice*: streets are horizontal, with even streets oriented East and odd streets oriented West, and avenues are vertical, with even avenues oriented North and odd avenues oriented South.

Definition 1. *A* taxi walk *is an oriented walk in $\mathbb{Z}^2$ that begins at the origin, never revisits a vertex, and never takes two left or two right turns in a row.*

We call these taxi walks because the violation of either restriction during a real taxi ride would cause suspicion among savvy passengers.

Lemma 3. *If an independent set I has a fault line F with no alternations, then it also has a fault line F' so that either F' or F'^R (the reversal of F') is a taxi walk.*

Proof: It is straightforward to see that if I has a fault line F with no alternation points, then it must have all of its edges oriented the same way (in $G_\diamond$) and it must be self-avoiding. Suppose F is a minimal length fault line in I without any alternations, and suppose that F has two successive turns. Because of the parity constraints, the vertices

immediately before and after these two turns must both connect edges that are in the same direction, and these five edges can be replaced by a single edge to form a shorter fault line without any alternations. This is a contradiction to F being minimal. □

The same argument shows that if F is a fault line with an alternation point, then there is a fault line that is the concatenation of two taxi walks (or the reversals of taxi walks). Lemma 3, and the extension just mentioned, are key ingredients in our proofs of both phase coexistence and slow mixing. They allow us to assume, as we do throughout, that all fault lines we work with are essentially taxi walks. For phase coexistence we will also need to understand the connection between Peierls contours and taxi walks.

Given an independent set I in $\mathbb{Z}^2$, let $(I^{\mathcal{O}})^+$ be the set of odd vertices in I together with their neighbors. Let R be any finite component of $(I^{\mathcal{O}})^+$, and let W be the unique infinite component of $\mathbb{Z}^2 \setminus R$. Let C be the complement of W (going from R to C essentially "fills in holes" in R). Finally, let γ be the set of edges with one end in W and one in C, and write $\gamma_\diamondsuit$ for the subgraph of $G_\diamondsuit$ induced by γ.

Lemma 4. *In $G_\diamondsuit$, $\gamma_\diamondsuit$ is a directed cycle that does not take two consecutive turns. Consequently, if an edge is removed from γ, the resulting path in $\mathbb{Z}^2$ (suitably translated and rotated) is a taxi walk.*

Proof: Because γ separates W from its complement, $\gamma_\diamondsuit$ must include a cycle surrounding a vertex of C, and since γ is in fact a *minimal* edge cutset (W and C are both connected), $\gamma_\diamondsuit$ must consist of just this cycle. To see both that $\gamma_\diamondsuit$ is correctly (i.e. uniformly) oriented in $G_\diamondsuit$, and that it does not take two consecutive turns, note that if either of these conditions were violated then we must have one of the following: a vertex of W (or C) all of whose neighbors are in C (or W), or a unit square in $\mathbb{Z}^2$ with both even vertices in C and both odd vertices in W (an easy case analysis). All of these situations lead to a 4-cycle in $\gamma_\diamondsuit$, a contradiction since $\gamma_\diamondsuit$ is a cycle whose length is evidently greater than 4 (in fact it must have length at least 12). □

A critical step in our arguments will be bounding the number of taxi walks. We start by recalling facts about standard self-avoiding walks (which have been studied extensively, although many basic questions remain; see, e.g., [15]). On $\mathbb{Z}^2$, the number c_n of walks of length n grows exponentially with n as $2^n \le c_n \le 4 \times 3^{n-1}$, since there are at most 3 ways to extend a self-avoiding walk of length $n-1$ and walks that only take steps to the right or up can always be extended in 2 ways. Hammersley and Welsh [11] showed that $c_n = \mu^n \exp(O(\sqrt{n}))$, where $\mu \approx 2.64$ is known as the *connective constant*. It is believed that $\exp(O(\sqrt{n}))$ here can be replaced by $\Theta(n^{11/32})$ (this is supported by considerable experimental and heuristical evidence).

Letting $\widetilde{c}_n$ be the number of taxi walks of length n, we quickly get $2^{n/2} < \widetilde{c}_n < 4 \times 3^{n-1}$. The upper bound here uses $\widetilde{c}_n < c_n$, and for the lower bound we observe that if we take two steps at a time in one direction we can always go East or North. With little extra work, we can make a significant improvement:

Lemma 5. *Let $\widetilde{c}_n$ be the number of taxi walks of length n. Then $\widetilde{c}_n = O((1+\sqrt{5})/2)^n$.*

Proof: At each vertex there are exactly two outgoing edges in $\mathbb{Z}^2$. If we arrive at v from u, then one of the outgoing edges continues the walk in the same direction and the other is a turn. The two allowable directions are determined by the parity of the

coordinates of v, so we can encode each walk as a bitstring $s \in \{0,1\}^{n-1}$. If $s_1 = 0$ then the walk starts by going East (along a street) and if $s_0 = 1$ the walk starts North along an avenue. For all $i > 1$, if $s_i = 0$ the walk continues in the same direction as the previous step, while if $s_i = 1$ then the walk turns in the permissible direction. In this encoding, the condition forbidding consecutive turns forces s to avoid having two 1's in a row, and hence $\widetilde{c}_n \le f_n = O(\phi^n)$, where f_n is the nth Fibonacci number and $\phi = (1+\sqrt{5})/2 \approx 1.618$ is the golden ratio. □

General considerations (discussed in Section 5) imply that there is a *taxi walk connective constant* $\mu_t > 0$ such that $\widetilde{c}_n = f_t(n)\mu_t^n$, where $f_t(n)$ grows sub-exponentially. A consequence of this is that

$$\text{if } \mu > \mu_t \text{ then for all large } n, \widetilde{c}_n < \mu^n. \tag{1}$$

Lemma 5 implies $\mu_t \le \phi$, and as we shall see from Theorems 4 and 5 below, this is enough to obtain phase coexistence, and slow mixing on the torus, for all $\lambda > \phi^4 - 1 \approx 5.85$. To obtain the stronger Theorems 1 and 2 we use more sophisticated tools, described in Section 5, to improve our bounds on $\widetilde{c}_n$.

Theorem 3. *We have* $1.5196 < \mu_t < 1.5884$ *and* $4.3332 < \mu_t^4 - 1 < 5.3646$.

3 Proof of Theorem 1 (phase coexistence on $\mathbb{Z}^2$ for large λ)

We work towards the following statement that implies Theorem 1 via Theorem 3.

Theorem 4. *The hard-core model on $\mathbb{Z}^2$ with activity λ admits multiple Gibbs states for all $\lambda > \mu_t^4 - 1$, where μ_t is the connective constant of taxi walks.*

We will not review the theory of Gibbs states, but just say informally that an interpretation of the existence of multiple Gibbs states is that the local behavior of a randomly chosen independent set in a box can be made to depend on a boundary condition, even in the limit as the size of the box grows to infinity. See e.g. [10] for a general treatment, or [4] for a treatment specific to the hard-core model on the lattice.

Let U_n be the box $[-n,+n]^2$, and I^{e} the independent set consisting of all even vertices of $\mathbb{Z}^2$. Let $\mathcal{J}_n^{\mathrm{e}}$ be the set of independent sets that agree with I^{e} off U_n, and μ_n^{e} the distribution supported on $\mathcal{J}_n^{\mathrm{e}}$ in which each set is selected with probability proportional to $\lambda^{|I \cap U_n|}$. Define μ_n^{o} analogously (with "even" everywhere replaced by "odd"). We will exhibit an event $\mathcal{A}$ with the property that for all large n, $\mu_n^{\mathrm{e}}(\mathcal{A}) \le 1/3$ and $\mu_n^{\mathrm{o}}(\mathcal{A}) \ge 2/3$. This is well known (see e.g. [4]) to be enough to establish existence of multiple Gibbs states.

The event $\mathcal{A}$ depends on a parameter $m = m(\lambda)$ whose value will be specified later. Specifically, $\mathcal{A}$ consists of all independent sets in $\mathbb{Z}^2$ whose restriction to U_m contains either an odd cross or a fault line. We will show that $\mu_n^{\mathrm{e}}(\mathcal{A}) \le 1/3$ for all sufficiently large n; reversing the roles of odd and even throughout, the same argument gives that under μ_n^{o} the probability of U_m having either an *even* cross or a fault line is also at most $1/3$, so that (by Lemma 1) $\mu_n^{\mathrm{o}}(\mathcal{A}) \ge 2/3$.

Write $\mathcal{A}_n^{\mathrm{e}}$ for $\mathcal{A} \cap \mathcal{J}_n^{\mathrm{e}}$; note that for all large n we have $\mu_n^{\mathrm{e}}(\mathcal{A}) = \mu_n^{\mathrm{e}}(\mathcal{A}_n^{\mathrm{e}})$. To show $\mu_n^{\mathrm{e}}(\mathcal{A}_n^{\mathrm{e}}) \le 1/3$ we will use the fact that $I \in \mathcal{A}_n^{\mathrm{e}}$ is in even phase (predominantly even-occupied) outside U_n, but because of either the odd cross or the fault line in U_m it is

not in even phase close to U_m; so there must be a contour marking the furthest extent of the even phase inside U_n. We will modify I inside the contour via a weight-increasing map, showing that an odd cross or fault line is unlikely.

3.1 The Contour and Its Properties

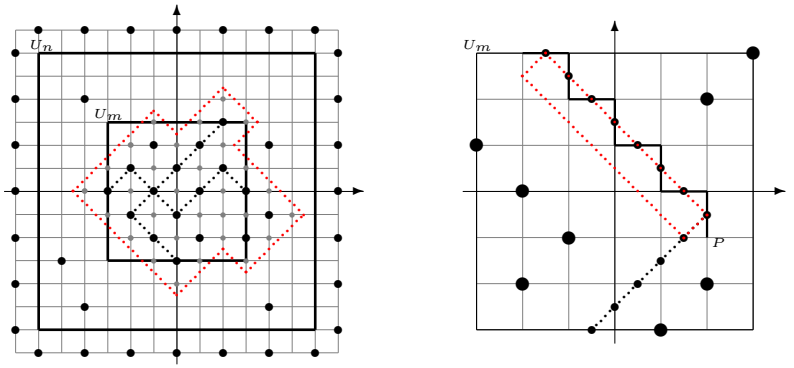
Fix $I \in \mathcal{A}_n^{\mathrm{e}}$. If I has an odd cross in U_m, we proceed as follows (using the notation from the discussion preceding Lemma 4). Let R be the component of $(I^{\mathcal{O}})^+$ that includes a particular odd cross. Note that because I agrees with I^{e} off U_n, R does not reach the boundary of U_n, and so as in the discussion preceding Lemma 4, we can associate to R a cutset γ separating it from the boundary of U_n.

Notice that γ is an edge cutset in U_n separating an interior connected region that meets U_m from an exterior connected region that includes the boundary of U_n, with all edges from the interior of γ to the exterior that all go from an unoccupied even vertex to an unoccupied odd vertex. This implies that $|\gamma|$, the number of edges in γ, is a multiple of 4, specifically four times the difference between the number of even and odd vertices in the interior of γ. Because the interior includes two points of the odd cross that are at distance at least $2m + 1$ from each other in U_m, we have a lower bound on γ that is linear in m; in particular, clearly $|\gamma| \geq m$. Note also that by Lemma 4, $\gamma_\Diamond$ must be a closed taxi walk. (See Fig. 3 (a).)

We now come to the heart of the Peierls argument. If we modify I by shifting it by one axis-parallel unit (positively or negatively) in the interior of γ and leaving it unchanged elsewhere, then the resulting set is still independent, and we may augment it with any vertex in the interior whose neighbor in the direction opposite to the shift is in the exterior. This is a straightforward verification; see [6, Lemma 6] or [9, Proposition 2.12] where this is proved in essentially the same setting. Furthermore, from [6, Lemma 5] each of the four possible shift directions free up exactly $|\gamma|/4$ vertices that can be added to the modified independent set.

We now describe the contour if I has a fault line in U_m. If there happens to be an odd occupied vertex in U_m then we construct γ as before, starting with some arbitrary component of $(I^{\mathcal{O}})^+$ that meets U_m in place of the component of an odd cross. If the resulting γ has a fault line in its interior, then γ and its associated $\gamma_\Diamond$ satisfy all the previously established properties immediately.

Otherwise, choose a fault line, which we can assume by Lemma 3 is a taxi walk or the concatenation of two taxi walks. Whether it has zero or one alternation points, we can find a path $P = u_1 u_2 \dots u_k$ in $\mathbb{Z}^2$ with k linear in m, with u_1 and u_k both odd, with no two consecutive edges parallel, and with the midpoints of the edges of the path inducing an alternation-free sub-path of the chosen fault line (essentially we are just taking a long piece of the fault line, on an appropriately chosen side of the alternation point, if there is one). This sub-path F_1 is a taxi walk. Next, we find a second path in $G_\Diamond$, disjoint from F_1, that always bisects completely unoccupied edges, and that taken together with F_1 completely encloses P. If there are no occupied odd vertices adjacent to even vertices of P, such a path is easy to find: we can shift F_1 one unit in an appropriate direction, and close off with an additional edge at each end (see Fig. 3 (b)). If there are some odd occupied vertices adjacent to some even vertices of P, then this translate of F_1 has to be looped around the corresponding components of $(I^{\mathcal{O}})^+$.



(a) (b)

Fig. 3. Independent sets in $\mathcal{A}_n^{\mathrm{e}}$ with (a) an odd cross in U_m with the corresponding C and $\gamma_\diamondsuit$ and (b) a fault line in U_m with the corresponding P and $\gamma_\diamondsuit$

Such a looping is possible because $(I^{\mathcal{O}})^+$ does not reach the boundary of U_n, nor does it enclose the fault line (if it did, we would be in the case of the previous paragraph).

This second path we have constructed may not be a taxi walk; however, following the proof of Lemma 3, we see that a *minimal* path F_2 satisfying the conditions of our constructed path is indeed a taxi walk. We take the concatenation of F_1 and F_2 to be $\gamma_\diamondsuit$ in this case, and take γ to be the set of edges that are bisected by vertices of $\gamma_\diamondsuit$. The contours in this case satisfy all the properties of those in the previous case. The standard strategies outlined in [6] and [9] can easily be used to derive the properties in this case. The one difference is that now $\gamma_\diamondsuit$ may not be a closed taxi walk; but at worst it is the concatenation of two taxi walks, both of length linear in m (and certainly it can be arranged that each has length at least $m/2$).

3.2 The Peierls Argument

For $J \in \mathcal{J}_n^{\mathrm{e}}$ set $w(J) = \lambda^{|J \cap U_n|}$. Our aim is to show that $w(\mathcal{A}_n^{\mathrm{e}})/w(\mathcal{J}_n^{\mathrm{e}}) \leq 1/3$. For $I \in \mathcal{A}_n^{\mathrm{e}}$, let $\varphi(I)$ be the set of independent sets obtained from I by shifting in the interior parallel to $(1,0)$ and adding all subsets of the $|\gamma|/4$ vertices by which the shifted independent set can be augmented. For $J \in \varphi(I)$, let S denote the set of added vertices. Define a bipartite graph on partite sets $\mathcal{A}_n^{\mathrm{e}}$ and $\mathcal{J}_n^{\mathrm{e}}$ by joining $I \in \mathcal{A}_n^{\mathrm{e}}$ to $J \in \mathcal{J}_n^{\mathrm{e}}$ if $J \in \varphi(I)$. Give edge IJ weight $w(I)\lambda^{|S|} = w(J)$ (where S is the set of vertices added to I to obtain J).

The sum of the weights of edges out of those $I \in \mathcal{A}_n^{\mathrm{e}}$ with $|\gamma(I)| = 4\ell$ is $(1+\lambda)^\ell$ times the sum of the weights of those I. For each $J \in \mathcal{J}_n^{\mathrm{e}}$, the sum of the weights of edges into J from this set of I's is $w(J)$ times the degree of J to the set. If $f(\ell)$ is a uniform upper bound on this degree, then

$$\frac{w(\mathcal{A}_n^{\mathrm{e}})}{w(\mathcal{J}_n^{\mathrm{e}})} \leq \sum_{\ell \geq m/4} \frac{f(\ell)}{(1+\lambda)^\ell}. \tag{2}$$

The lower bound on ℓ here is crucial. The standard Peierls argument takes $\mathcal{A}$ to be the event that a fixed vertex is occupied, and the analysis of probabilities associated with

this event requires dealing with short contours, leading to much weaker bounds than we are able to obtain.

To control $f(\ell)$, observe that for each $J \in \mathcal{J}_n^{\mathrm{e}}$ and contour γ of length 4ℓ there is at most one I with $\gamma(I) = \gamma$ such that $J \in \varphi(I)$ (I can be reconstructed from J and γ, since the set S of added vertices can easily be identified; cf. [9, Section 2.5]). It follows that we may bound $f(\ell)$ by the number of contours of length 4ℓ with a vertex of U_m in their interiors.

Fix $\mu > \mu_t$. By the properties of contours we have established, up to translations of contours this number is at most the maximum of $\mu^{4\ell}$ and $\sum_{j+k=4\ell:\ j,k\geq m/2} \mu^j \mu^k = 4\ell\mu^{4\ell}$ (for all large m, here using (1). The restriction of $G_\diamondsuit$ to U_m has at most $4(2m+1)^2 \leq 17m^2$ edges, so there are at most this many translates of a contour that can have a vertex of U_m in its interior. We may bound $f(\ell)$ by $68m^2\ell\mu^{4\ell}$ and so the sum in (2) by $\sum_{\ell\geq m} 68m^2\ell(\mu^4/(1+\lambda))^\ell$. For any fixed $\lambda > \mu^4 - 1$, there is an m large enough so that this sum is at most $1/3$; we take any such m to be $m(\lambda)$, completing the proof.

4 Proof of Theorem 2 (slow mixing of Glauber dynamics)

Let $G \subset \mathbb{Z}^2$ be an $n \times n$ lattice region and let Ω be the set of independent sets on G. Our goal is to sample from Ω according to the Gibbs distribution, where each $I \in \Omega$ is assigned probability $\pi(I) = \lambda^{|I|}/Z$, where $Z = \sum_{I'\in\Omega} \lambda^{|I'|}$. *Glauber dynamics* is a local Markov chain that connects two independent sets if they have symmetric difference of size one. The Metropolis probabilities [16] that force the chain to converge to the Gibbs distribution are given by

$$P(I,I') = \begin{cases} \frac{1}{2n}\min\left(1, \lambda^{|I'|-|I|}\right), & \text{if } I \oplus I' = 1, \\ 1 - \sum_{J\sim I} P(I,J), & \text{if } I = I', \\ 0, & \text{otherwise.} \end{cases}$$

The conductance, introduced by Jerrum and Sinclair [21], is a good measure of a chain's mixing rate. Let

$$\Phi = \min_{S\in\Omega:\pi(S)\leq 1/2} \frac{\sum_{x\in S, y\notin S} \pi(x)P(x,y)}{\pi(S)},$$

where $\pi(S) = \sum_{x\in S} \pi(x)$. From [21] we know that $\frac{\Phi^2}{2} \leq Gap(P) \leq 2\Phi$, where $Gap(P)$ is the spectral gap of the transition matrix. The spectral gap is well-known to be a measure of the mixing rate of a Markov chain, so a partition of the state space witnessing exponentially small conductance is sufficient to show slow mixing. The machinery of Section 2 provides such a partition.

We are now ready to present the proof of slow mixing, starting first with the two-dimensional torus. In the interest of space, we defer the proof of the second part of Theorem 2 showing slow mixing on regions with free boundary conditions for the full version of the paper. The proof is similar to the argument described in [19] and utilizing the improvements given here for the torus.

Let n be even, and let $\widehat{G} = \{0,\ldots,n-1\}\times\{0,\ldots,n-1\}$ be the $n\times n$ lattice region with toroidal boundary conditions. Let $\widehat{\Omega}$ be the set of independent sets on $\widehat{G}$, and $\widehat{\pi}$ the

Gibbs distribution. Lemma 2 shows that $\widehat{\Omega}$ may be partitioned into $\widehat{\Omega}_{\mathcal{F}}$ (independent sets with a fault), $\widehat{\Omega}_0$ (independent sets with an even cross) and $\widehat{\Omega}_1$ (independent sets with an odd cross), and that furthermore $\widehat{\Omega}_0$ and $\widehat{\Omega}_1$ are not directly connected by moves in the chain. It remains to show that $\widehat{\pi}(\widehat{\Omega}_{\mathcal{F}})$ is exponentially smaller than both $\widehat{\pi}(\widehat{\Omega}_0)$ and $\widehat{\pi}(\widehat{\Omega}_1)$. (Clearly $\widehat{\pi}(\widehat{\Omega}_0) = \widehat{\pi}(\widehat{\Omega}_1)$ by symmetry.) Notice that on the torus we may assume that fault lines have no alternation points; since they start and end at the same place, the number of alternation points must be even.

For an independent set $I \in \widehat{\Omega}_{\mathcal{F}}$ with fault $F = (F_1, F_2)$, partition I into two sets, I_A and I_B, depending on which side of F_1 and F_2 they lie. Define the length of F to be the number of edges (in $G_\diamondsuit$) on F_1 and F_2. Note that if F_1 has no alternation points then it has length $N = 2n + 2\ell$ for some positive integer ℓ.[2]

Let $I' = \sigma(I, F)$ be the configuration formed by shifting I_A one to the right. Let $F_1' = \sigma(F_1)$ and $F_2' = \sigma(F_2)$ be the images of the fault under this shift. We define the points that lie in $F_1 \cap F_1'$ and $F_2 \cap F_2'$ to be the points that fall "in between" F and $F' := (F_1', F_2')$. It will be convenient to order the set of possible fault lines so that given a configuration $I \in \widehat{\Omega}_{\mathcal{F}}$ we can identify its *first* fault. The following results are modified from [19] and rely on the new characterization of faults as taxi walks.

Lemma 6. *Let $\widehat{\Omega}_F$ be the configurations in $\widehat{\Omega}_{\mathcal{F}}$ with first fault $F = (F_1, F_2)$. Write the length of F as $4n + 4\ell$. Then $\pi(\widehat{\Omega}_F) \le (1+\lambda)^{-(n+\ell)}$.*

Proof: We define an injection $\phi_F : \widehat{\Omega}_F \times \{0,1\}^{n+\ell} \hookrightarrow \Omega$ so that $\widehat{\pi}(\phi_F(I, r)) = \widehat{\pi}(I)\lambda^{|r|}$. The injection is formed by cutting the torus $\widehat{G}$ along F_1 and F_2 and shifting one of the two connected pieces in any direction by one unit. There will be exactly $n+\ell$ unoccupied points near F that are guaranteed to have only unoccupied neighbors. We add a subset of the vertices in this set to I according to bits that are one in the vector r. Given this map, we have

$$1 = \widehat{\pi}(\widehat{\Omega}) \ge \sum_{I \in \widehat{\Omega}_F} \sum_{r \in \{0,1\}^{n+\ell}} \widehat{\pi}(\phi_F(I, r)) = \sum_{I \in \widehat{\Omega}_F} \widehat{\pi}(I) \sum_{r \in \{0,1\}^{n+\ell}} \lambda^{|r|}.$$

□

Theorem 5. *Let $\widehat{\Omega}$ be the set of independent sets on $\widehat{G}$ weighted by $\widehat{\pi}(I) = \lambda^{|I|}/Z$, where $Z = \sum_{I \in \widehat{\Omega}} \lambda^{|I|}$. Let $\Omega_{\mathcal{F}}$ be the set of independent sets on $\widehat{G}$ with a fault. Then for any $\lambda > \mu_t^4 - 1$, there is a constant $c > 0$ such that $\widehat{\pi}(\Omega_{\mathcal{F}}) \le e^{-cn}$.*

Proof: Fix μ satisfying $\lambda > \mu^4 - 1 > \mu_t^4 - 1$, where μ_t is the taxi walk connective constant. Summing over locations for the two faults F_1 and F_2 and using Lemma 6,

$$\widehat{\pi}(\widehat{\Omega}_{\mathcal{F}}) \;=\; \sum_F \widehat{\pi}(\widehat{\Omega}_F) \;\le\; \sum_F (1+\lambda)^{-(n+\ell)} \;\le\; n^2 \sum_{i=0}^{(n^2-2n)/2} \left(\frac{\mu^4}{1+\lambda}\right)^{n+i}.$$

[2] In [19, Section 2.2] this is erroneously presented as $N = n + 2\ell$, and the missing factor 2 remains absent for all the remaining calculations; the corrected calculations lead to the weaker bounds quoted in the introduction.

The second inequality here uses Theorem 3. By our choice of μ we get (for large n) $\pi(\Omega_{\mathcal{F}}) \leq e^{-cn}$ for some constant $c > 0$; and we can easily modify this constant to deal with all smaller values of n. □

From Theorem 3, we know that $\mu_t^4 - 1 < 5.3646$. From Theorem 5, we thus get the first part of Theorem 2, as well as the following stronger result.

Corollary 1. *Fix $\lambda > \mu_t^4 - 1$. Glauber dynamics for sampling independent sets on the $n \times n$ torus $\widehat{G}$ takes time at least e^{cn} to mix, for some constant $c > 0$ (depending on λ).*

Proof: We will bound the conductance by considering $S = \widehat{\Omega}_0$. It is clear that $\widehat{\pi}(S) \leq 1/2$ since $\overline{S} = \widehat{\Omega}_{\mathcal{F}} \cup \widehat{\Omega}_1$ and $\widehat{\pi}(\widehat{\Omega}_0) = \widehat{\pi}(\widehat{\Omega}_1)$. Thus,

$$\Phi \leq \frac{\sum_{s \in \widehat{\Omega}_0, t \in \widehat{\Omega}_{\mathcal{F}}} \widehat{\pi}(s) P(s,t)}{\widehat{\pi}(\widehat{\Omega}_0)} = \frac{\sum_{s \in \widehat{\Omega}_0, t \in \widehat{\Omega}_{\mathcal{F}}} \widehat{\pi}(t) P(t,s)}{\widehat{\pi}(\widehat{\Omega}_0)} \leq \frac{\sum_{t \in \widehat{\Omega}_{\mathcal{F}}} \widehat{\pi}(t)}{\widehat{\pi}(\widehat{\Omega}_0)} = \frac{\widehat{\pi}(\widehat{\Omega}_{\mathcal{F}})}{\widehat{\pi}(\widehat{\Omega}_0)}.$$

Given Theorem 5, it is trivial to show that $\widehat{\pi}(\widehat{\Omega}_0) > 1/3$, thereby establishing that the conductance is exponentially small. It follows that Glauber dynamics takes exponential time to converge. □

5 Taxi Walks: Bounds and Limits

We conclude by justifying the upper bound on the number of taxi walks given in Theorem 3, as well as providing a lower bound on μ_t. It is necessary to first establish the submultiplicativity of $\widetilde{c}_n$ (or, equivalently, the subadditivity of $\log \widetilde{c}_n$).

Lemma 7. *Let $\widetilde{c}_n$ be the number of taxi walks of length n and let $1 \leq i \leq n-1$. Then $\widetilde{c}_n \leq \widetilde{c}_i \, \widetilde{c}_{n-i}$.*

Proof: As with traditional self-avoiding walks, the key is to recognize that if we split a taxi walk of length n into two pieces, the resulting pieces are both self-avoiding. Let $s = s_1, \ldots, s_n$ be a taxi walk of length n and let $1 \leq i \leq n-1$. Then the initial segment of the walk $s_I = s_1, \ldots, s_{i+1}$ is a taxi walk of length i. Let $p = (x, y)$ be the ith vertex of the walk s. Let s_F be the final $n - i$ steps of the walk s starting at p. We define $f(s_F)$ by translating the walk so that $f(p)$ is the origin, reflecting horizontally if p_x is odd and reflecting vertically if p_y is odd. Notice that this always produces a valid taxi walk of length $n - i$ and the map f is invertible given p. Therefore $\widetilde{c}_n \leq \widetilde{c}_i \, \widetilde{c}_{n-i}$. □

It follows from Lemma 7 that $a_n = \log \widetilde{c}_n$ is subadditive, i.e., $a_{n+m} \leq a_n + a_m$. By Fekete's Lemma (see, e.g., [23, Lemma 1.2.2]) we know $\lim_{n \to \infty} a_n/n$ exists and

$$\lim_{n \to \infty} \frac{a_n}{n} = \inf \frac{a_n}{n}. \tag{3}$$

Thus, we can write the number of taxi walks as $\widetilde{c}_n = \mu_t^n f_t(n)$, where μ_t is the connective constant associated with taxi walks and $f_t(n)$ is subexponential in n.

Subadditivity gives us a strategy for getting a better bound on μ_t. From (3) we see that for all n, $\log \widetilde{c}_n / n$ is an upper bound for $\log \mu_t$. We exactly enumerated taxi walks of length n, for $n \leq 60$; see `http://nd.edu/~dgalvin1/TD/` for this and other

data. Using $c_{60} = 2189670407434$ gives a bound of $\mu_t < 1.6058$. Note that exact counts for larger n will immediately improve our bounds on both μ_t and λ_c.

The connective constant for ordinary self-avoiding walks has been well studied, and some of the methods used to obtain bounds there can be adapted to deal with taxi walks. In particular, a method of Alm [1] is useful. Fix $n > m > 0$. Construct a square matrix $A(m,n)$ whose ij entry counts the number of taxi walks of length n that begin with the ith taxi walk of length m, and end with the jth taxi walk of length m, for some fixed ordering of the walks of length m. To make sense of this, it is necessary to choose, for each $v \in \mathbb{Z}^2$, an orientation preserving map f_v of $\mathbb{Z}^2$ that sends the origin to v; saying that a walk of length n ends with the jth walk of length m means that if the length m terminal segment of the walk is transformed by f_v^{-1} to start at the origin, where v is the first vertex of the terminal segment, then the result is the jth walk of length m. Then a theorem of Alm [1] may be modified to show that μ_t is bounded above by $\lambda_1(A(m,n))^{1/(n-m)}$, where λ_1 indicates the largest positive eigenvalue. (Note that when $m = 0$ this recovers the subadditivity bound discussed earlier).

We have calculated $A(20,60)$. This is a square matrix of dimension 20114, and a simple symmetry argument reduces the dimension by a factor of 2. Using MATLAB, we could estimate the largest eigenvalue to obtain $\mu_t < 1.5884$ and $\mu_t^4 - 1 < 5.3646$.

A similar strategy can be used to derive lower bounds on μ_t in order to determine the theoretical limitations of our approach of characterizing contours by taxi walks. We have already given the trivial lower bound $\mu_t \geq \sqrt{2}$. To improve this, we consider *bridges* (introduced for ordinary self-avoiding walks by Kesten [12]). A *bridge*, for our purposes, is a taxi walk that begins by moving from the origin $(0,0)$ to the point $(1,0)$, never revisits the y-axis, and ends by taking a step parallel to the x-axis to a point on the walk that has maximum x-coordinate over all points in the walk (but note that this maximum does not have to be uniquely achieved at the final point).

Let b_n be the number of bridges of length n. Then bridges are supermultiplicative, i.e., $b_n \geq b_i b_{n-i}$ (and $\log b_n$ is superadditive). To see this, note that if β_1 and β_2 are bridges, then they both begin and end at vertices whose y-coordinates are even because they are taking steps to the East. If the parities of the x-coordinates of the first vertices in β_1 and β_2 agree, then the concatenation of β_1 and an appropriate translation of β_2 is also a bridge; if the parities are different then concatenation of β_1 with a translation of β_2 after reflecting horizontally will be a valid bridge. Notice that the parity of the x-coordinate of the two pieces allows us to recover whether a reflection was necessary to keep the walk on the directed Manhattan lattice, so bridges are indeed supermultiplicative. It similarly follows that there are at least b_n^k taxi walks of length kn (just concatenate k length n bridges), so that

$$\mu_t = \lim_{m\to\infty} \widetilde{c}_m^{1/m} \geq \lim_{k\to\infty} (b_n^k)^{1/nk} = b_n^{1/n}.$$

We have enumerated bridges of length up to 60, in particular discovering that $b_{60} = 80312795498$, leading to $\mu_t > 1.5196$ and $\mu_t^4 - 1 > 4.3332$.

A consequence of our lower bound on μ_t is that our approach to phase coexistence cannot give anything better than $\lambda_c \leq 4.3332$; this tells us that new ideas will be needed to reach the value of 3.796 suggested by computations as the true value of λ_c.

References

1. Alm, S.: Upper bounds for the connective constant of self-avoiding walks. Combinatorics, Probability & Computing 2, 115–136 (1993)
2. Baxter, R., Entig, I., Tsang, S.: Hard-square lattice gas. J. Stat. Phys. 22, 465–489 (1980)
3. Beffara, V., Duminil-Copin, H.: The self-dual point of the two-dimensional random-cluster model is critical for $q \geq 1$. Probability Theory and Related Fields 153, 511–542 (2012)
4. van den Berg, J., Steif, J.E.: Percolation and the hard-core lattice model. Stochastic Processes and their Applications 49, 179–197 (1994)
5. Borgs, C.: Personal communication
6. Borgs, C., Chayes, J.T., Frieze, A., Kim, J.H., Tetali, P., Vigoda, E., Vu, V.H.: Torpid mixing of some MCMC algorithms in statistical physics. In: Proc. 40th IEEE Symp. on Foundations of Computer Science (FOCS), pp. 218–229 (1999)
7. Brightwell, G.R., Häggström, O., Winkler, P.: Non-monotonic behavior in hard-core and Widom-Rowlinson models. J. Stat. Phys. 94, 415–435 (1999)
8. Dobrushin, R.L.: The problem of uniqueness of a Gibbs random field and the problem of phase transitions. Functional Analysis and its Applic. 2, 302–312 (1968)
9. Galvin, D., Kahn, J.: On phase transitions in the hard-core model on Z^d. Combinatorics, Probability & Computing 13, 137–164 (2004)
10. Georgii, H.-O.: Gibbs Measures and Phase Transitions. de Gruyter, Berlin (1988)
11. Hammersley, J.M., Welsh, D.J.A.: Further results on the rate of convergence to the connective constant of the hyper cubic lattice. Quarterly J. of Mathematics 2, 108–110 (1962)
12. Kesten, H.: On the number of self-avoiding walks. J. Math. Phys. 4, 960–969 (1963)
13. Lubezky, E., Sly, A.: Critical Ising on the square lattice mixes in polynomial time. To appear in Communications in Mathematical Physics
14. Luby, M., Vigoda, E.: Fast convergence of the Glauber dynamics for sampling independent sets. Random Structures & Algorithms 15, 229–241 (1999)
15. Madras, N., Slade, G.: The Self-Avoiding Walk. Birkhäuser, Boston (1993)
16. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. J. of Chemical Physics 21, 1087–1092 (1953)
17. Onsager, L.: Crystal statistics. I. A two-dimensional model with an order-disorder transition. Physics Review Letters 65, 117–149 (1944)
18. Radulescu, D.C., Styer, D.F.: The Dobrushin-Shlosman phase uniqueness criterion and applications to hard squares. J. Stat. Phys. 49, 281–295 (1987)
19. Randall, D.: Slow mixing of Glauber dynamics via topological obstructions. In: Proc. 17th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 870–879 (2006)
20. Restrepo, R., Shin, J., Tetali, P., Vigoda, E., Yang, L.: Improved mixing condition on the grid for counting and sampling independent sets. In: Proc. 52nd IEEE Symp. on Foundations of Computer Science (FOCS), pp. 140–149 (2011)
21. Sinclair, A.J., Jerrum, M.R.: Approximate counting, uniform generation and rapidly mixing Markov chains. Information and Computation 82, 93–133 (1989)
22. Sly, A.: Computational Transition at the Uniqueness Threshold. In: Proc. 51st IEEE Symp. on Foundations of Computer Science (FOCS), pp. 287–296 (2010)
23. Steele, J.M.: Probability Theory and Combinatorial Optimization. SIAM (1997)
24. Vera, J.C., Vigoda, E., Yang, L.: Improved bounds on the phase transition for the hard-core model in 2-dimensions. In: Raghavendra, P., Raskhodnikova, S., Jansen, K., Rolim, J.D.P. (eds.) APPROX/RANDOM 2013. LNCS, vol. 8096, pp. 705–719. Springer, Heidelberg (2013)
25. Weitz, D.: Counting independent sets up to the tree threshold. In: Proc. 38th ACM Symp. on the Theory of Computing (STOC), pp. 140–149 (2006)

Fast Private Data Release Algorithms for Sparse Queries

Avrim Blum* and Aaron Roth**

[1] Carnegie Mellon University
avrim@cs.cmu.edu
[2] University of Pennsylvania
aaroth@cis.upenn.edu

Abstract. We revisit the problem of accurately answering large classes of statistical queries while preserving differential privacy. Previous approaches to this problem have either been very general but have not had run-time polynomial in the size of the database, have applied only to very limited classes of queries, or have relaxed the notion of worst-case error guarantees. In this paper we consider the large class of *sparse* queries, which take non-zero values on only polynomially many universe elements. We give efficient query release algorithms for this class, in both the interactive and the non-interactive setting. Our algorithms also achieve better accuracy bounds than previous general techniques do when applied to sparse queries: our bounds are independent of the universe size. In fact, even the runtime of our interactive mechanism is independent of the universe size, and so can be implemented in the "infinite universe" model in which no finite universe need be specified by the data curator.

1 Introduction

A database $\mathcal{D}$ represents a finite collection of individual records from some *data universe* $\mathcal{X}$, which represents the set of all *possible* records. We typically think of $\mathcal{X}$ as being extremely large: exponentially large in the size of the database, or in some cases, possibly even infinite. A fundamental task in private data analysis is to accurately answer statistical queries about a database $\mathcal{D}$, while provably preserving the privacy of the individuals whose records are contained in $\mathcal{D}$. The privacy solution concept we use in this paper is *differential privacy*, which has become standard, and which we define in section 2.

Accurately answering statistical (i.e. linear or counting) queries is the most well studied problem in differential privacy, and the results to date come in two types. There are a large number of extremely general and powerful techniques (see for example [BLR08, DNR+09, DRV10, RR10, HT10, HR10]) that can accurately answer arbitrary families of statistical queries which can be exponentially large in the size of the database. Unfortunately, these techniques all

* Department of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213.
** Department of Computer and Information Science, University of Pennsylvania, Philadelphia PA 19104. This research was partially supported by an NSF CAREER Grant and NSF grant CNS-1065060.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 395–410, 2013.

have running time that is at least linear in the size of the data universe $|\mathcal{X}|$ (i.e. possibly *exponential* in the size of the database), and so are in many cases impractical. There are also several techniques that do run in polynomial time, but that are limited: either they can answer queries from a very general and structurally rich class (i.e. all low-sensitivity queries), but can only answer a linear number of such queries (i.e. [DMNS06]), or they can answer a very large number of queries, but only from a structurally very simple class (i.e. intervals on the unit line[1] [BLR08]), or as in several recent results (for conjunction and parity queries respectively) [GHRU11, HRS12] they run in polynomial time, but offer only average case guarantees for randomly chosen queries. One of the main open questions in data privacy is to develop general data release techniques comparable in power to the known exponential time techniques that run in polynomial time. There is evidence, however, that this is not possible for arbitrary linear queries [DNR+09, UV11, GHRU11].

In this paper, we consider a restricted but structurally rich class of linear queries which we call *sparse* queries. We say that a query is m-sparse if it takes non-zero values on only m universe elements, and that a class of queries is m-sparse if each query it contains is m' sparse for some $m' \leq m$. We will typically think of m as being some polynomial in the database size n. Note that although each individual query is restricted to have support on only a polynomially sized subset of the data universe, different queries in the same class can have different supports, and so a class of sparse queries can still have support over the entire data universe. This is what prevents previous algorithms from being made efficient, simply by running them on a restricted universe: it may not be possible to consider a restricted universe for a set of sparse queries, because every universe element may take positive value on some query in a sparse class! Note that the class of m-sparse queries is both very large (of size roughly $|\mathcal{X}|^m$), and very structurally complex (the class of m-sparse queries have VC-dimension m). Sparse queries represent questions about individuals whose answer is rarely "yes" when asked about an individual who is drawn uniformly at random from the data population. Nevertheless, such questions can be useful to a data analyst who has some knowledge about which segment of the population a database might be drawn from. For example, a database resulting from a medical study might contain individuals who have some rare disease, but the data analyst does not know *which* disease – although there may be many such queries, each one is sparse. Alternately, a data analyst might have knowledge about the participants of several previous studies, and might want to know how much overlap there is between the participants of each previous study and of the current study. In general, sparse queries will only be useful to a data analyst who has some knowledge about the database, beyond that it is merely a subset of an exponentially sized data universe. Our results can therefore be viewed as a way of privately releasing information about a database that is useful to specialists – but is privacy preserving no matter who makes use of it. In general, this work can be thought

[1] The algorithm of [BLR08] can be generalized to answer axis-aligned rectangle queries in constant dimension, but this is still a class that has only constant VC-dimension.

of as part of an agenda to find ways to make use of the *domain knowledge* of the data analyst, to make private analysis of large-scale data-sets feasible.

1.1 Results

We give two algorithms for releasing accurate answers to m-sparse queries while preserving differential privacy: one in the interactive setting, in which the data curator acts as an intermediary and must answer an adaptively chosen stream of queries as they arrive, and one in the non-interactive setting, in which the data curator must in one shot output a data-structure which encodes the answers to every query of interest. In the interactive setting, we require that the running time needed to answer each query is bounded by a polynomial in n, the database size (so to answer any sequence of k queries takes time $k \cdot \text{poly}(n)$). In the non-interactive setting, the entire computation must be performed in time polynomial in n, and the time required to evaluate any query on the output data structure must also be polynomial. Therefore, from the point of view of running time, the non-interactive setting is strictly more difficult than the interactive setting.

In the interactive setting, we give the following utility bound:

Theorem 1 (Informal, some parameters hidden). *There exists an (ϵ, δ)-differentially private query release mechanism in the interactive setting, with running time per query $\tilde{O}(m/\alpha^2)$ that is α-accurate with respect to any set of k adaptively chosen m-sparse queries with:*

$$\alpha = O\left(\frac{(\log m)^{1/4}\left(\log \frac{1}{\delta} \log k\right)^{1/2}}{(\epsilon n)^{1/2}}\right)$$

In the non-interactive setting, we give the bound:

Theorem 2 (Informal, some parameters hidden). *There exists an (ϵ, δ)-differentially private query release mechanism in the non-interactive setting, with running time polynomial in the database size n, m, and $\log |\mathcal{X}|$, that is α-accurate with respect to any class of k m-sparse linear queries, with:*

$$\alpha = \tilde{O}\left(\log k \frac{\sqrt{m \log\left(\frac{1}{\delta}\right)}}{\epsilon n}\right)$$

Several aspects of these theorems are notable. First, the accuracy bounds do not have any dependence on the size of the data universe $|\mathcal{X}|$, and instead depend only on the sparsity parameter m. Therefore, in addition to efficiency improvements, these results give accuracy improvements for sparse queries, when compared to the general purpose (inefficient) mechanisms for linear queries, which typically have accuracy which depends on $\log |\mathcal{X}|$. Since we typically view $|\mathcal{X}|$ as exponentially large in the database size, whereas m is only polynomially large

in the database size for these algorithms to be efficient, this can be a large improvement in accuracy.

Second, the interactive mechanism does not even have a dependence on $|\mathcal{X}|$ in its running time! In fact, it works even in an *infinite* universe (e.g. data entries with string valued attributes without pre-specified upper bound on length)[2]. In this setting, queries may still be concisely specified as a list of polynomially many individuals from the possibly infinite universe that satisfy the query. Moreover, because the accuracy of this mechanism depends only very mildly on m, and the running time is linear in m, it can be used to answer m-sparse queries for arbitrarily large polynomial values of m, where the mechanism is constrained only by the available computational resources.

The non-interactive mechanism in contrast has a worse dependence on m. This bound essentially matches the error that would result from releasing the perturbed *histogram* of the database, but does so in a way that requires computation and output representation only polynomial in n (rather than linear in $|X|$, as releasing a histogram would require). Because accuracy bounds > 1 are trivial, this mechanism only guarantees non-trivial accuracy for m-sparse queries with $m << n^2/\log k$ (This is still of course a very large class of queries: there are roughly $|\mathcal{X}|^{n^2/\log k}$ such queries, i.e., super-exponentially many in n). Nevertheless, there are distinct advantages to having a non-interactive mechanism that only needs to be run once. This is among the first *polynomial time* non-interactive mechanisms for answering an exponentially large, unstructured class of queries while preserving differential privacy.

We note that our results give as a corollary, more efficient algorithms for answering conjunctions with many literals. This complements the beautiful recent work of Hardt, Rothblum, and Servedio [HRS12], who give more efficient algorithms for answering conjunctions with few literals, based on reductions to threshold learning problems.

1.2 Techniques

Our interactive mechanism is a modification of the very general multiplicative weights mechanism of Hardt and Rothblum [HR10]. We give the interactive mechanism via the framework of [GRU12] which efficiently maps objects called *iterative database constructions* (defined in section 3) into private query release mechanisms in the interactive setting. IDC algorithms are very similar to online learning algorithms in the mistake bound model, and we use this analogy to implement a version of the multiplicative weights IDC of Hardt and Rothblum [HR10] analogously to how the Winnow algorithm is implemented in the *infinite*

[2] The algorithm must be able to read a *name* for each universe element it deals with, and so it can of course not deal with elements that have no finite description length. But for a (countably) infinite universe, the running time would depend on the length of the largest string used to denote a universe element encountered during the running of the algorithm, and not in any a-priori way on the (unboundedly large) size of the universe.

attribute model of learning, defined by Blum [Blu90]. The algorithm roughly works as follows: the multiplicative weights algorithm normally maintains a distribution over $|\mathcal{X}|$ elements, one for each element in the data universe. It can be easily implemented in such a way so that when it is updated after a query Q arrives, only those weights corresponding to elements in the support of the query Q are updated: for an m-sparse query, this means it only need update m positions. It also comes with a guarantee that it never needs to perform more than $\log |\mathcal{X}|/\alpha^2$ updates before achieving error α, and so at most $m \log |\mathcal{X}|/\alpha^2$ elements ever need to be updated. The key insight is to pick a smaller universe, $\widehat{\mathcal{X}}$, such that $\widehat{\mathcal{X}} \geq m \log \widehat{\mathcal{X}}/\alpha^2$, but *not to commit to the identity of the elements in this universe* before running the algorithm, letting all elements be initially unassigned. The algorithm then maintains a hash table mapping elements of $\mathcal{X}$ to elements of $\widehat{\mathcal{X}}$. Elements in $\mathcal{X}$ are assigned temporary mappings to elements in $\widehat{\mathcal{X}}$ as queries come in, but are only assigned permanent mappings when an update is performed. Because only $\log \widehat{\mathcal{X}}/\alpha^2$ updates are ever performed, and $\widehat{\mathcal{X}}$ was chosen such that $\widehat{\mathcal{X}} \geq m \log \widehat{\mathcal{X}}/\alpha^2$, the algorithm never runs out of elements of $\widehat{\mathcal{X}}$ to permanently assign. Because $|\widehat{\mathcal{X}}|$ depends only on the desired accuracy α and the sparsity parameter m, and *not* on $\mathcal{X}$ in any way, the algorithm can be implemented and run without any knowledge of $\mathcal{X}$ (even for infinite universes), and neither the running time nor the resulting accuracy depend on $|\mathcal{X}|$. We emphasize that the advantage of this approach over an alternative technique of running multiplicative weights on a small random projection of the data universe, is that our approach works even with *adaptively chosen queries*. Hashing the data-universe to a smaller size using standard methods would no longer permit guarantees for queries that may be adaptively chosen.

The non-interactive mechanism releases a random projection of the database into polynomially many dimensions, together with the corresponding projection matrix. Queries are evaluated by computing their projection using the public projection matrix, and then taking the inner product of the projected query and the projected database. The difficulty comes because the projection matrix projects vectors from $|\mathcal{X}|$-dimensional space to poly(n) dimensional space, and so normally would take $|\mathcal{X}|$poly(n)-many bits to represent. Our algorithms are constrained to run in time poly(n), however, and so we need a concise representation of the projection matrix. We achieve this by using a matrix implicitly generated by a family of limited-independence hash functions which have concise representations. This requires using a limited independence version of the Johnson-Lindenstrauss lemma, and of concentration bounds. This algorithm also gives accuracy bounds which are independent of $|\mathcal{X}|$.

1.3 Related Work

Differential privacy was introduced by Dwork, McSherry, Nissim, and Smith [DMNS06], and has since become the standard solution concept for privacy in the theoretical computer science literature. There is now a vast literature concerning differential privacy, so we mention here only the most relevant work, without attempting to be exhaustive. Dwork et al. [DMNS06] also introduced the *Laplace*

mechanism, which is able to efficiently answer arbitrary low-sensitivity queries in the interactive setting. The Laplace mechanism does not make efficient use of the *privacy budget* however, and can answer only linearly many queries in the database size.

Blum, Ligett, and Roth [BLR08] showed that in the non-interactive setting, it is possible to answer *exponentially* sized families of counting queries. This result was extended and improved by Dwork et al. [DNR+09] and Dwork, Rothblum, and Vadhan [DRV10], who gave improved running time and accuracy bounds, and for (ϵ, δ)-differential privacy gave similar results for arbitrary low sensitivity queries. Roth and Roughgarden [RR10] showed that accuracy bounds comparable to [BLR08] could be achieved even in the *interactive* setting, and this result was improved in both accuracy and running time by Hardt and Rothblum, who give the multiplicative weights mechanism, which achieves nearly optimal accuracy and running time [HR10]. Gupta, Roth, and Ullman [GRU12] generalize the algorithms of [RR10, HR10] into a generic framework in which objects called *iterative database constructions* efficiently reduce to private data release mechanisms in the interactive setting. Unfortunately, the running time of all of the algorithms discussed here is at least linear in $|\mathcal{X}|$, and so typically exponential in the size of the private database. Moreover, there are both computational and information theoretic lower bounds suggesting that it may be very difficult to give private release algorithms for generic linear queries with substantially better run time [DNR+09, UV11, GHRU11]. As in this work, these algorithms give a guarantee on the worst-case error of any answered query.

There is also a small body of work giving more efficient query release mechanisms for specific classes of queries. [BLR08] gave an efficient (running time polynomial in the database size n) algorithm for releasing the answers for 1-dimensional intervals on the discretized unit-line in the non-interactive setting. As far as we know, prior to this work, this was the only efficient mechanism in either the interactive or non-interactive settings for releasing the answers to an exponentially sized family of queries with worst-case error. This class is however structurally very simple: it has VC-dimension only 2. Other efficient algorithms relax the notion of utility, no longer guaranteeing worst-case error for all queries. [BLR08] also give an efficient algorithm for releasing *halfspace* queries in the unit sphere, but this algorithm only guaranteed accurate answers for halfspaces that happened to have large *margin* with respect to the points in the database. Gupta et al [GHRU11] gave an algorithm for releasing *conjunctions* over d attributes to *average* error α over any product distribution (over conjunctions), which runs in time $d^{O(1/\alpha)}$. This was improved to have running time $O(d^{\log 1/\alpha})$ by Cheraghchi et al. [CKKL12]. Note that these algorithms only run in polynomial time for constant values of α, and only give accuracy bounds in expectation over random queries. Recently, Hardt, Rothblum, and Servedio [HRS12] gave an algorithm for releasing conjunctions defined on k out of d literals with an average-error guarantee *for any* pre-specified distribution in time $d^{\tilde{O}(\sqrt{k})}$. Using the private boosting algorithm of [DRV10], they leverage this result to give an algorithm for releasing k-literal conjunctions with worst-case error guarantees, which increases

the running time to $d^{\tilde{O}(k)}$, although still only requiring databases of size $d^{\tilde{O}(\sqrt{k})}$. They also gave an efficient (i.e. running time polynomial in n) algorithm for releasing *parity* queries to low average error over product distributions. We remark that our results give a complementary bound for large conjunctions (with a better sample complexity requirement). Our online algorithm can release all conjunctions on $d - k$ out of d literals with worst-case error guarantees in time $d^{\tilde{O}(k)}$, requiring databases of size only $\tilde{O}(k^{1.5} \log d)$.

The efficient interactive mechanism we give in section 3 is based on an analogy between iterative database construction (IDC) algorithms and online learning algorithms in the mistake bound model. We implement the multiplicative weights IDC of Hardt and Rothblum [HR10] analogously to how Winnow is implemented in the *infinite attribute model* of Blum [Blu90]. In our setting, it can be thought of as an *infinite universe model* that has no dependence on the universe size in either the running time or accuracy bounds. This involves running the multiplicative weights algorithm on a much smaller universe. Hardt and Rothblum [HR10] also gave a version of their algorithm which ran on a small subset of the universe to give efficient run-time guarantees. The main difference is that we select the subset of the universe that we run the multiplicative weights algorithm on adaptively, based on the queries that arrive, whereas [HR10] select the subset nonadaptively, independently of the queries. [HR10] give average case utility bounds for linear queries on randomly selected databases; in contrast, we give worst-case utility bounds that hold for all input databases, but only for sparse linear queries.

The efficient non-interactive mechanism we give in section 4 is based on random projections using families of limited independence hash functions, which have previously been used for space-bounded computations in the streaming model [CW09, KN10]. Limited independence hash functions have also previously been used for streaming algorithms in the context of differential privacy [DNP+10].

2 Preliminaries

A database $\mathcal{D}$ is a multiset of elements from some (possibly infinite) abstract universe $\mathcal{X}$. We write $|\mathcal{D}| = n$ to denote the cardinality of $\mathcal{D}$. For any $x \in \mathcal{X}$ we can also write $D[x]$ to denote: $\mathcal{D}[x] = |\{x' \in \mathcal{D} : x' = x\}|$ the number of elements of type x in the database. Viewed this way, a database $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$ is a vector with integer entries in the range $[0, n]$.

A linear query $Q : \mathcal{X} \to [0, 1]$ is a function mapping elements in the universe to values on the real unit interval. For notational convenience, we will define $Q(\emptyset) = 0$. We can also evaluate a linear query on a database. The value of a linear query Q on a database is simply the average value of Q on elements of the database:

$$Q(\mathcal{D}) = \frac{1}{n} \sum_{x \in \mathcal{D}} Q(x) = \frac{1}{n} \sum_{x \in \mathcal{X}} Q(x) D[x]$$

Similarly to how we can think of a database as a vector, we can think of a query as a vector $Q \in [0,1]^{|\mathcal{X}|}$ with $Q[x] = Q(x)$. Viewed this way, $Q(\mathcal{D}) = \frac{1}{n}\langle Q, \mathcal{D}\rangle$.

It will sometimes be convenient to think of normalized databases (with entries that sum to 1). For a database $\mathcal{D}$ of size n, we define the corresponding normalized database $\hat{\mathcal{D}}$ to be the database such that $\hat{\mathcal{D}}[x] = \mathcal{D}[x]/n$. We evaluate a linear query on a normalized database by computing $Q(\hat{\mathcal{D}}) = \sum_{x\in\mathcal{X}} Q(x)\hat{\mathcal{D}}[x] = \langle Q, \hat{\mathcal{D}}\rangle$. Note that $Q(\mathcal{D}) = Q(\hat{\mathcal{D}})$.

Definition 1 (Sparsity). *The* sparsity *of a linear query Q is $|\{x \in \mathcal{X} : Q(x) > 0\}|$, the number of elements in the universe on which it takes a non-zero value. We say that a query is m-sparse if its sparsity is at most m. We will also refer to the class of all m-sparse linear queries, denoted $\mathcal{Q}_m$.*

In this paper, we will assume that given an m-sparse query, we can quickly (in time polynomial in m) enumerate the elements $x \in \mathcal{X}$ on which $Q(x) > 0$.

Remark 1. While the assumption that we can quickly enumerate the non-zero values of a query may not always hold, it is indeed the case that for many natural classes of queries, we can enumerate the non-zero elements in time *linear* in m. For example, this holds for queries that are specified as lists of the universe elements on which the query is non-zero, as well as for many implicitly defined query classes such as conjunctions, disjunctions, parities, etc.[3] Of course, classes like conjunctions are typically not sparse, but conjunctions with $d - O(\log n)$ literals are, and their support can be quickly enumerated (even though there are superpolynomially many such conjunctions).

2.1 Utility

We will design algorithms which can accurately answer large numbers of sparse linear queries. We will be interested in both *interactive* mechanisms and *non-interactive* mechanisms. A non-interactive mechanism takes as input a database, runs one time, and outputs some data structure capable of answering many queries without further interaction with the data release mechanism. An interactive mechanism takes as input a stream of queries, and must provide a numeric answer to each query before the next one arrives.

Definition 2 (Accuracy for non-Interactive Mechanisms). *Let $\mathcal{Q}$ be a set of queries. A non-interactive mechanism $M : \mathcal{X}^* \to R$ for some abstract range R is* (α, β)-accurate for $\mathcal{Q}$ *if there exists a function* $\mathrm{Eval} : \mathcal{Q} \times R \to \mathbb{R}$ *s.t. for every database $\mathcal{D} \in \mathcal{X}^*$, with probability at least $1 - \beta$ over the coins of M, $M(\mathcal{D})$ outputs $r \in R$ such that* $\max_{Q\in\mathcal{Q}} |Q(\mathcal{D}) - \mathrm{Eval}(Q, r)| \leq \alpha$. *We will abuse notation and write $Q(r) = \mathrm{Eval}(Q, r)$.*

[3] The set of conjunctions over the d-dimensional boolean hypercube with $d - log(n)$ literals are n-sparse. Even though there are superpolynomially many such conjunctions, it is simple to enumerate the entries on which these conjunctions take non-zero value in time linear in n. We can simply enumerate all of the $2^{\log n} = n$ values that the unassigned variables can take.

M is efficient *if both M and* Eval *run in time polynomial in the size of the database n.*

Definition 3 (Accuracy for Interactive Mechanisms). *Let $\mathcal{Q}$ be a set of queries. An interactive mechanism M takes as input an adaptively chosen stream of queries $Q_1, \ldots, Q_k \in \mathcal{Q}$ and for each query Q_i, outputs an answer $a_i \in \mathbb{R}$ before receiving Q_{i+1}. It is* (α, β)-accurate *if for every database $\mathcal{D} \in \mathcal{X}^*$, with probability at least $1 - \beta$ over the coins of M: $\max_i |Q_i - a_i| \leq \alpha$.*

M is efficient *if the update time for each query (i.e. the time to produce answer a_i after receiving query Q_i) is polynomial in the size of the database n.*

2.2 Differential Privacy

We will require that our algorithms satisfy *differential privacy*, defined as follows. We must first define the notion of *neighboring databases.*

Definition 4 (Neighboring Databases). *Two databases $\mathcal{D}, \mathcal{D}'$ are* neighbors *if they differ only in the data of a single individual: i.e. if their symmetric difference is $|\mathcal{D} \triangle \mathcal{D}'| \leq 1$.*

Definition 5 (Differential Privacy [DMNS06]). *A randomized algorithm M acting on databases and outputting elements from some abstract range R is (ϵ, δ)-differentially private if for all pairs of neighboring databases $\mathcal{D}, \mathcal{D}'$ and for all subsets of the range $S \subseteq R$ the following holds:*

$$\Pr[M(\mathcal{D}) \in S] \leq \exp(\epsilon) \Pr[M(\mathcal{D}') \in S] + \delta$$

Remark 2. For a non-interactive mechanism, R is simply the set of data-structures that the mechanism outputs. For an interactive mechanism, because the queries may be adaptively chosen by an adversary, R is the set of query/answer transcripts produced by the algorithm when interacting with an arbitrary adversary. For a detailed treatment of differential privacy and adaptive adversaries, see [DRV10].

Additional preliminaries can be found in the full version [BR11].

3 A Fast IDC Algorithm for Sparse Queries

In this section we use the abstraction of an *iterative database construction* that was introduced by Gupta, Roth, and Ullman [GRU12]. It was shown in [GRU12] that efficient IDC algorithms automatically reduce to efficient differentially private query release mechanisms in the interactive setting. Roughly, an IDC mechanism works by maintaining a sequence of data structures $\mathcal{D}_1, \mathcal{D}_2, \ldots$ that give increasingly good approximations to the input database $\mathcal{D}$ (in a sense that depends on the IDC). Moreover, these mechanisms produce the next data structure in the sequence by considering only one query Q that *distinguishes* the real database in the sense that $Q(\mathcal{D}_t)$ differs significantly from $Q(\mathcal{D})$.

Syntactically, we will consider functions of the form $\mathbf{U} : \mathcal{R}_{\mathbf{U}} \times \mathcal{Q} \times \mathbb{R} \to \mathcal{R}_{\mathbf{U}}$. The inputs to $\mathbf{U}$ are a data structure in $\mathcal{R}_{\mathbf{U}}$, which represents the current data structure $\mathcal{D}_t$; a query Q, which represents the distinguishing query, and may be restricted to a certain set $\mathcal{Q}$; and also a real number which estimates $Q(\mathcal{D})$. Formally, we define a *database update sequence* , to capture the sequence of inputs to $\mathbf{U}$ used to generate the database sequence $\mathcal{D}_1, \mathcal{D}_2, \ldots$.

Definition 6 (Database Update Sequence). *Let $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$ be any database and let $\left\{(\mathcal{D}_t, Q_t, \widehat{A}_t)\right\}_{t=1,\ldots,T} \in (\mathcal{R}_{\mathbf{U}} \times \mathcal{Q} \times \mathbb{R})^T$ be a sequence of tuples. We say the sequence is an $(\mathbf{U}, \mathcal{D}, \mathcal{Q}, \alpha, T)$*-database update sequence *if it satisfies the following properties:*

1. $\mathcal{D}_1 = \mathbf{U}(\emptyset, \cdot, \cdot)$,
2. *for every* $t = 1, 2, \ldots, T$, $|Q_t(\mathcal{D}) - Q_t(\mathcal{D}_t)| \geq \alpha$,
3. *for every* $t = 1, 2, \ldots, T$, $\left|Q_t(\mathcal{D}) - \widehat{A}_t\right| < \alpha$,
4. *and for every* $t = 1, 2, \ldots, T-1$, $\mathcal{D}_{t+1} = \mathbf{U}(\mathcal{D}_t, Q_t, \widehat{A}_t)$.

Definition 7 (Iterative Database Construction). *Let $\mathbf{U} : \mathcal{R}_{\mathbf{U}} \times \mathcal{Q} \times \mathbb{R} \to \mathcal{R}_{\mathbf{U}}$ be an update rule and let $B : \mathbb{R} \to \mathbb{R}$ be a function. We say $\mathbf{U}$ is a $B(\alpha)$-*iterative database construction for query class $\mathcal{Q}$ *if for every database $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$, every $(\mathbf{U}, \mathcal{D}, \mathcal{Q}, \alpha, T)$-database update sequence satisfies $T \leq B(\alpha)$.*

Note that the definition of an $B(\alpha)$-iterative database construction implies that if $\mathbf{U}$ is a $B(\alpha)$-iterative database construction, then given any maximal $(\mathbf{U}, \mathcal{D}, \mathcal{Q}, \alpha, T)$-database update sequence, the final database $\mathcal{D}_T$ must satisfy $\max_{Q \in \mathcal{Q}} |Q(\mathcal{D}) - Q(\mathcal{D}_T)| \leq \alpha$ or else there would exist another query satisfying property 2 of Definition 6, and thus there would exist a $(\mathbf{U}, \mathcal{D}, \mathcal{Q}, \alpha, T+1)$-database update sequence, contradicting maximality.

$B(\alpha)$-IDC algorithms generically reduce to (ϵ, δ)-differentially private (α, β)-accurate query release mechanisms in an efficiency preserving way. This framework was implicitly used by [RR10] and [HR10].

Theorem 3 ([GRU12]). *If there exists a $B(\alpha)$-IDC algorithm for a class of queries $\mathcal{Q}$ using a class of datastructures $\mathcal{R}_{\mathbf{U}}$ that take time at most $p(n, \alpha, |\mathcal{X}|)$ to update their hypotheses, and time at most $q(n, \alpha, |\mathcal{X}|)$ to evaluate a query on any $\mathcal{D} \in \mathcal{R}_{\mathbf{U}}$, then for any $0 < \epsilon, \delta, \beta < 1$ there exists an (ϵ, δ)-differentially private query release mechanism in the interactive setting that has update time at most $O(p(n, \alpha, \mathcal{X}) + q(n, \alpha, \mathcal{X}))$ and is (α, β)-accurate with respect to any adaptively chosen sequence of k queries from $\mathcal{Q}$ where α is the solution to the following equality: $\alpha = \frac{3000\sqrt{B(\alpha)}\log(4/\delta)\log(k/\beta)}{\epsilon n}$*

In this section we will give an efficient IDC algorithm for the class of m-sparse queries, and then call on Theorem 3 to reduce it to a differentially private query release mechanism in the interactive setting.

First we introduce the Sparse Multiplicative Weights data structure, which will be the class of datastructures $\mathcal{R}_{\mathbf{U}}$ that the Sparse Multiplicative Weights IDC algorithm uses.:

Definition 8 (Sparse Multiplicative Weights Data Structure). *The sparse multiplicative weights data structure $\mathcal{D}^{SMW}$ of size s is composed of three parts. We write $\mathcal{D}^{SMW} = (\mathcal{D}, h, ind)$.*

1. *$\mathcal{D}$ is a collection of s real valued variables $x_1, \ldots, x_s$, with $x_i \in [0,1]$ for all $i \in [s]$. Variable x_i for $i \in [s]$ is referenced by $\mathcal{D}[i]$. Initially $x_i = 1/s$ for all $i \in [s]$. We define $\mathcal{D}[i] = 0$ for all $i > s$.*
2. *h is a hash function $h : \mathcal{X} \to [s] \cup \emptyset$ mapping elements in the universe X to indices $i \in [s]$. Elements $x \in \mathcal{X}$ can also be unassigned in which case we write $h(x) = \emptyset$. Initially, $h(x) = \emptyset$ for all $x \in \mathcal{X}$ We write $h^{-1}(i) = x$ if $h(x) = i$, and $h^{-1}(i) = \emptyset$ if there does not exist any $x \in \mathcal{X}$ such that $h(x) = i$.*
3. *$ind \in [s+1]$ is a counter denoting the index of the first unassigned variable. For all $i < ind$, there exists some $x \in \mathcal{X}$ such that $h(x) = i$. For all $i \geq ind$, there does not exist any $x \in \mathcal{X}$ such that $h(x) = i$. Initially $ind = 1$.*

If $ind \leq s$, we can add *an unassigned element $x \in \mathcal{X}$ to $\mathcal{D}^{SMW}$. Adding an element $x \in \mathcal{X}$ to $\mathcal{D}^{SMW}$ sets $h(x) \leftarrow ind$ and increments $ind \leftarrow ind + 1$. If $ind = s + 1$, attempting to add an element causes the data structure to report* ***FAILURE****.*

A linear query Q is evaluated on a sparse MW data structure $\mathcal{D}^{SMW} = (\mathcal{D}, h)$ as follows.

$$Q(\mathcal{D}^{SMW}) = \sum_{x \in \mathcal{X} : Q(x) > 0 \wedge h(x) \neq \emptyset} Q(x) \cdot \mathcal{D}[h(x)] + \sum_{x \in \mathcal{X} : Q(x) > 0 \wedge h(x) = \emptyset} Q(x) \cdot \mathcal{D}[ind]$$

We now present Algorithm 3, the Sparse Multiplicative Weights (SMW) IDC algorithm for m-sparse queries. The algorithm is a version of the Hardt/Rothblum Multiplicative Weights IDC [HR10], modified to work without any dependence on the universe size. It will run multiplicative weights update steps over the variables of the SMW data structure, using the SMW data structure to delay assigning variables to particular universe elements $x \in \mathcal{X}$ until necessary. Note that it is not simply running the multiplicative weights algorithm from [HR10] implicitly: doing so would yield guarantees that depend on the cardinality of the universe $|\mathcal{X}|$. Instead, the guarantees we will get will depend only on m, and so will carry over even to the infinite-universe setting.

Theorem 4. *The Sparse Multiplicative Weights algorithm is a $B(\alpha)$-IDC for the class of m-sparse queries $\mathcal{Q}_m$, where:*

$$B(\alpha) = 4\frac{\log s + 1}{\alpha^2}$$

and s is the smallest integer such that $s/(\log(s) + 1) \geq 4m/\alpha^2$.

The analysis largely follows the Multiplicative Weights analysis given by Hardt and Rothblum [HR10]. The main difference is that rather than using one global potential function, we must use a different potential function for each database update sequence, defined as a function of the state of the hash table in the last SMW datastructure in the sequence. We must also argue that we never run

SMW$(\mathcal{D}_t^{\text{SMW}} = (\mathcal{D}_t, h_t, \text{ind}_t), Q_t, \widehat{A}_t)$:

1. **If** $\mathcal{D}_t^{\text{SMW}} = \emptyset$
   (a) **Let** s be the smallest integer such that $s/(\log(s)+1) \geq 4m/\alpha^2$.
   (b) **Return** a new Sparse MW data structure $\mathcal{D}_1^{\text{SMW}} = (D_1, h_1, \text{ind}_1)$ of size s with $h_1(x) = \emptyset$ for all $x \in \mathcal{X}$, $x_i = 1/s$ for all $i \in [s]$, and $\text{ind}_1 = 1$.
2. **Let** $\mathcal{D}_{t+1}^{\text{SMW}} = (\mathcal{D}_{t+1}, h_{t+1}, \text{ind}_{t+1}) \leftarrow \mathcal{D}_t^{\text{SMW}}$
3. **Update:** For all $x \in \mathcal{X}$ such that $Q_t(x) > 0$: **If** $h_{t+1}(x) = \emptyset$ then **add** x to $\mathcal{D}_{t+1}^{\text{SMW}}$.
4. **If** $\widehat{A}_t < Q_t(\mathcal{D}_t^{\text{SMW}})$ **Then Update:** For all $x \in \mathcal{X}$ such that $Q_t(x) > 0$: Let $\mathcal{D}_{t+1}[h_{t+1}(x)] \leftarrow \mathcal{D}_{t+1}[h_{t+1}(x)] \cdot \exp(-\eta Q_t(x))$
5. **Else Update:** For all $x \in \mathcal{X}$ such that $Q_t(x) > 0$: Let $\mathcal{D}_{t+1}[h_{t+1}(x)] \leftarrow \mathcal{D}_{t+1}[h_{t+1}(x)] \cdot \exp(\eta Q_t(x))$
6. **Normalize:** For all $i \in [s]$: $\mathcal{D}_{t+1}[i] = \frac{\mathcal{D}_{t+1}[i]}{\sum_{j=1}^{s} \mathcal{D}_{t+1}[j]}$
7. **Output** $\mathcal{D}_{t+1}^{\text{SMW}}$.

Fig. 1. The Sparse Multiplicative Weights (SMW) IDC Algorithm for m-sparse queries, adapted from the MW IDC of [HR10]. It is instantiated with an accuracy parameter $\eta = \alpha/2$. It takes as input a sparse MW datastructure $\mathcal{D}^{\text{SMW}}$, an m-sparse query $Q \in \mathcal{Q}_m$, and an estimate of the query value $\widehat{A}$.

out of variables to assign in the SMW data structure, which would cause it to return **FAILURE**. To argue this, we apply the technique of Blum Hellerstein and Littlestone [BHL95], used to adapt Winnow to the infinite attribute model. The proof appears in the full version.

Finally, we may observe that both the update time for the SMW IDC and the time to evaluate a query on the SMW datatructure is $O(s) = \tilde{O}(m/\alpha^2)$. Therefore, we may instantiate Theorem 3 with the SMW IDC algorithm to obtain the main result of this section:

Theorem 5. *For any $0 < \epsilon, \delta, \beta < 1$ there exists an (ϵ, δ)-differentially private query release mechanism in the interactive setting, with running time per query $\tilde{O}(m/\alpha^2)$ that is (α, β)-accurate with respect to the set of all m-sparse linear queries $\mathcal{Q}_m$, with:*

$$\alpha = O\left(\frac{(\log m)^{1/4}\left(\log\frac{4}{\delta}\log\frac{k}{\beta}\right)^{1/2}}{(\epsilon \cdot n)^{1/2}}\right)$$

Proof. The proof follows by instantiating Theorem 3 with the SMW IDC algorithm, together with the bound $B(\alpha) = \frac{4(\log s+1)}{\alpha^2}$ proven in Theorem 4, and recalling that s is the smallest integer such that $s/(\log s + 1) \geq 4m/\alpha^2$.

3.1 Applications to Conjunctions

In this section, we briefly mention a simple application of this algorithm to the problem of releasing conjunctions with many literals. The algorithm given in

this section leads to new results for releasing conjunctions on $d-k$ out of d literals. This complements the recent results of Hardt, Rothblum, and Servedio [HRS12] for releasing conjunctions on k out of d literals. The class of conjunctions are defined over the universe $\mathcal{X} = \{0,1\}^d$ equal to the d-dimensional boolean hypercube.

Definition 9. *A conjunction is a linear query specified by a subset of variables $S \subseteq [d]$, and defined by the predicate $Q_S : \{0,1\}^d \to \{0,1\}$ where $Q_S(x) = \prod_{i \in S} x_i$. We say that a conjunction Q_S has t literals if $|S| = t$.*

Remark 3. The set of all conjunctions of $d-k$ literals, denoted C_{d-k} is 2^k sparse, and of size $|C| \leq d^k$.

We can release the answers to all queries in C_{d-k} by running the sparse multiplicative weights algorithm on each query. We therefore get the following corollary:

Corollary 1. *There exists an (ϵ, δ)-differentially private algorithm in the non-interactive release setting with running time at most*

$$\tilde{O}\left(|C_{d-k}| \cdot \frac{2^k}{\alpha^2}\right) = \tilde{O}\left(\frac{(2d)^k}{\alpha^2}\right)$$

that is (α, β)-accurate for the set of all conjunctions on $d-k$ literals, which requires a database of size only:

$$n \geq \frac{k^{1.5} \log \frac{1}{\delta} \log \frac{d}{\beta}}{\epsilon \alpha^2}$$

We note that the running time of this algorithm is comparable to the running time of the algorithm of [HRS12] for releasing all conjunctions of k out of d literals to worst case error (time roughly $\tilde{O}(|C_k|) = \tilde{O}(d^k)$), but requires a database of size only roughly $k^{1.5} \log d$, rather than $d^{\tilde{O}(\sqrt{k})}$ as required by [HRS12]. Of course, conjunctions on k literals are a more natural class than conjunctions on $d-k$ literals, but the results are complementary.

Moreover, applying the sparse multiplicative weights algorithm in the interactive setting gives polynomially bounded running time per query for conjunctions on $d-k$ literals for any $k = O(\log n)$. Note that this is still a super-polynomially sized class of conjunctions, with $|C_{O(\log n)}| = d^{O(\log n)}$. This is the first interactive query release algorithm that we are aware of that is simultaneously privacy-efficient and computationally-efficient for a super-polynomially sized class of conjunctions (or any other family of queries with super-constant VC-dimension).

4 A Non-interactive Mechanism via Random Projection

In this section, we give a non-interactive query release mechanism for sparse queries based on releasing a perturbed random projection of the private database, together with the projection matrix. Note that when viewing the database $\mathcal{D}$ as a

vector, it is an $|\mathcal{X}|$-dimensional object: $\mathcal{D} \in \mathbb{R}^{|\mathcal{X}|}$. A linear projection of $\mathcal{D}$ into T dimensions is obtained by multiplying it by a $|\mathcal{X}| \times T$ matrix, which cannot even be represented explicitly if we require algorithms that run in time polynomial in $n = |\mathcal{D}|$ for $n << |\mathcal{X}|$. It is therefore essential that we use projection matrices which can be represented concisely using hash functions drawn from limited-independence families.

We will use a limited-independence version of the Johnson-Lindenstrauss lemma presented in [KN10], first proven by [Ach01, CW09].

Theorem 6 (The Johnson-Lindenstrauss Lemma with Limited Independence [Ach01, CW09, KN10]). *For $d > 0$ an integer and any $0 < \varsigma, \tau < 1/2$, let A be a $T \times d$ random matrix with $\pm 1/\sqrt{T}$ entries that are r-wise independent for $T \geq 4 \cdot 64^2 \varsigma^{-2} \log(1/\tau)$ and $r \geq 2\log(1/\tau)$. Then for any $x \in \mathbb{R}^d$: $\Pr_A[|||Ax||_2^2 - ||x||_2^2| \geq \varsigma ||x||_2^2] \leq \tau$*

We will use the fact that random projections also preserve pairwise inner products. The following corollary is well known:

Corollary 2. *For $d > 0$ an integer and any $0 < \varsigma, \tau < 1/2$, let A be a $T \times d$ random matrix with $\pm 1/\sqrt{T}$ entries that are r-wise independent for $T \geq 4 \cdot 64^2 \varsigma^{-2} \log(1/\tau)$ and $r \geq 2\log(1/\tau)$. Then for any $x, y \in \mathbb{R}^d$: $\Pr_A[|\langle (Ax), (Ay) \rangle - \langle x, y \rangle| \geq \frac{\varsigma}{2}(||x||_2^2 + ||y||_2^2)] \leq 2\tau$*

Definition 10 (Random Projection Data Structure). *The random projection datastructure $\mathcal{D}_r$ of size T is composed of two parts: we write $\mathcal{D}_r = (u, f)$.*

1. *$u \in \mathbb{R}^T$ is a vector of length T.*
2. *$f : [|\mathcal{X}| \cdot T] \to \{-1/\sqrt{T}, 1/\sqrt{T}\}$ is a hash function implicitly representing a $T \times |\mathcal{X}|$ projection matrix $A \in \{-1/\sqrt{T}, 1/\sqrt{T}\}^{T \times |\mathcal{X}|}$. For any $(i, j) \in T \times |\mathcal{X}|$, we write $A[i, j]$ for $f(|\mathcal{X}| \cdot (i-1) + j)$.*

To evaluate a linear query Q on a random projection datastructure $\mathcal{D}_r = (u, f)$ we first project the query and then evaluate the projected query. To project the query we compute a vector $\widehat{Q} \in \mathbb{R}^T$ as follows. For each $i \in [T]$ $\widehat{Q}[i] = \sum_{x \in \mathcal{X} : Q(x) > 0} Q[x] \cdot A[i, x]$ Then we output: $Q(\mathcal{D}_r) = \frac{1}{n}\langle \widehat{Q}, u \rangle$.

Theorem 7. *SparseProject is (ϵ, δ)-differentially private.*

Theorem 8. *For any $0 < \epsilon, \delta < 1$, and any $\beta < 1$, and with respect to any class of m-sparse linear queries $\mathcal{Q} \subset \mathcal{Q}_m$ of cardinality $|\mathcal{Q}| \leq k$, SparseProject is (α, β)-accurate for: $\alpha = \tilde{O}\left(\log\left(\frac{k}{\beta}\right) \frac{\sqrt{m \log\left(\frac{1}{\delta}\right)}}{\epsilon n}\right)$ where the $\tilde{O}$ hides a term logarithmic in $(m + n)$.*

4.1 Applications to Conjunctions

In this section, we again briefly briefly mention a simple application of our non-interactive mechanism to the problem of releasing conjunctions with many literals. This gives the first polynomial time algorithm for non-interactively releasing a super-polynomially sized set of conjunctions.

SparseProject$(\mathcal{D}, \epsilon, \delta, \beta, m, k)$

1. **Let** $\tau \leftarrow \frac{\beta}{4k}$, $T \leftarrow 4 \cdot 64^2 \cdot \log\left(\frac{1}{\tau}\right)\left(\frac{m^{3/2}}{2} + \frac{n^4}{2\sqrt{m}} + \sqrt{m}n^2\right)$, $\sigma \leftarrow \frac{\epsilon}{\sqrt{8\ln(1/\delta)}}$
2. **Let** f be a randomly chosen hash function from a family of $2\log(kT/2\beta)$-wise independent hash functions mapping $[T \times |\mathcal{X}|] \to \{-1/\sqrt{T}, 1/\sqrt{T}\}$. Write $A[i,j]$ to denote $f(|\mathcal{X}| \cdot (i-1) + j)$.
3. **Let** $u, \nu \in \mathbb{R}^T$ be a vectors of length T.
4. **For** $i = 1$ to T
   (a) **Let** $u_i \leftarrow \sum_{x:\mathcal{D}[x]>0} \mathcal{D}[x] \cdot A[i,x]$
   (b) **Let** $\nu_i \leftarrow \mathrm{Lap}(1/\sigma)$
5. **Output** $\mathcal{D}_r = (u + \nu, f)$.

Fig. 2. SparseProject takes as input a private database $\mathcal{D}$ of size n, privacy parameters ϵ and δ, a confidence parameter β, a sparsity parameter m, and the size of the target query class k

Remark 4. The set of all conjunctions of $d-k$ literals, denoted C_{d-k} is 2^k sparse, and of size $|C_{d-k}| \leq d^k$.

Sparseproject therefore gives the following corollary:

Corollary 3. *There exists an (ϵ, δ)-differentially private algorithm in the non-interactive release setting with polynomially bounded running time, that is (α, β)-accurate for the class of conjunctions $C_{d-\log n}$ on $d - \log n$ literals for:* $\alpha = \tilde{O}\left(\left(\log n \log d + \log \frac{1}{\beta}\right) \frac{\sqrt{\log\left(\frac{1}{\delta}\right)}}{\epsilon\sqrt{n}}\right)$

Note that $C_{d-\log n}$ is a super-polynomially sized set of conjunctions. As far as we know, this represents the first algorithm in the non-interactive setting with non-trivial accuracy guarantees for a super-polynomially sized set of conjunctions that also achieves polynomial running time.

References

[Ach01] Achlioptas, D.: Database-friendly random projections. In: Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, p. 281. ACM (2001)

[BHL95] Blum, A., Hellerstein, L., Littlestone, N.: Learning in the presence of finitely or infinitely many irrelevant attributes. JCSS 50(1), 32–40 (1995)

[BLR08] Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 609–618. ACM (2008)

[Blu90] Blum, A.: Learning boolean functions in an infinite attribute space. In: Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing, pp. 64–72. ACM (1990)

[BR11] Blum, A., Roth, A.: Fast private data release algorithms for sparse queries. arXiv preprint arXiv:1111.6842 (2011)

[CKKL12] Cheraghchi, M., Klivans, A., Kothari, P., Lee, H.K.: Submodular functions are noise stable. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1586–1592. SIAM (2012)

[CW09] Clarkson, K.L., Woodruff, D.P.: Numerical linear algebra in the streaming model. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, pp. 205–214. ACM (2009)

[DMNS06] Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)

[DNP+10] Dwork, C., Naor, M., Pitassi, T., Rothblum, G.N., Yekhanin, S.: Pan-private streaming algorithms. In: Proceedings of ICS (2010)

[DNR+09] Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: Proceedings of the 41st Annual ACM Symposium on the Theory of Computing, pp. 381–390. ACM, New York (2009)

[DRV10] Dwork, C., Rothblum, G.N., Vadhan, S.: Boosting and differential privacy. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, pp. 51–60. IEEE (2010)

[GHRU11] Gupta, A., Hardt, M., Roth, A., Ullman, J.: Privately Releasing Conjunctions and the Statistical Query Barrier. In: Proceedings of the 43rd Annual ACM Symposium on the Theory of Computing. ACM, New York (2011)

[GRU12] Gupta, A., Roth, A., Ullman, J.: Iterative constructions and private data release. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 339–356. Springer, Heidelberg (2012)

[HR10] Hardt, M., Rothblum, G.N.: A multiplicative weights mechanism for privacy-preserving data analysis. In: 51st Annual IEEE Symposium on Foundations of Computer Science, pp. 61–70. IEEE (2010)

[HRS12] Hardt, M., Rothblum, G.N., Servedio, R.A.: Private data release via learning thresholds. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 168–187. SIAM (2012)

[HT10] Hardt, M., Talwar, K.: On the Geometry of Differential Privacy. In: The 42nd ACM Symposium on the Theory of Computing, STOC 2010 (2010)

[KN10] Kane, D.M., Nelson, J.: A derandomized sparse johnson-lindenstrauss transform. arXiv preprint arXiv:1006.3585 (2010)

[RR10] Roth, A., Roughgarden, T.: Interactive Privacy via the Median Mechanism. In: The 42nd ACM Symposium on the Theory of Computing, STOC 2010 (2010)

[UV11] Ullman, J., Vadhan, S.: PCPs and the hardness of generating private synthetic data. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 400–416. Springer, Heidelberg (2011)

Local Reconstructors and Tolerant Testers for Connectivity and Diameter

Andrea Campagna[1,⋆], Alan Guo[2,⋆⋆], and Ronitt Rubinfeld[2,3,⋆⋆⋆]

[1] IT University of Copenhagen
acam@itu.dk
[2] MIT
aguo@mit.edu
[3] Tel Aviv University
ronitt@csail.mit.edu

Abstract. A local property reconstructor for a graph property is an algorithm which, given oracle access to the adjacency list of a graph that is "close" to having the property, provides oracle access to the adjacency matrix of a "correction" of the graph, i.e. a graph which has the property and is close to the given graph. For this model, we achieve local property reconstructors for the properties of connectivity and k-connectivity in undirected graphs, and the property of strong connectivity in directed graphs. Along the way, we present a method of transforming a local reconstructor (which acts as a "adjacency matrix oracle" for the corrected graph) into an "adjacency list oracle". This allows us to recursively use our local reconstructor for $(k-1)$-connectivity to obtain a local reconstructor for k-connectivity.

We also extend this notion of local reconstruction to parametrized graph properties (for instance, having diameter at most D for some parameter D) and require that the corrected graph has the property with parameter close to the original. We obtain a local reconstructor for the low diameter property, where if the original graph is close to having diameter D, then the corrected graph has diameter roughly $2D$.

We also exploit a connection between local property reconstruction and property testing, observed by Brakerski, to obtain new tolerant property testers for all of the aforementioned properties. Except for the one for connectivity, these are the first tolerant property testers for these properties.

1 Introduction

Suppose we are given a very large graph G that is promised to be close to having a property $\mathcal{P}_s$. For example, $\mathcal{P}_D$ might denote the property of having diameter

⋆ This research was done while at IT University of Copenhagen and while visiting the Blavatnik School of Computer Science of Tel Aviv University.

⋆⋆ Research supported in part by NSF grants CCF-0829672, CCF-1065125, CCF-6922462, NSF-STC Award 0939370, and an NSF Graduate Research Fellowship.

⋆⋆⋆ Research supported by NSF grants CCF-1065125 and CCF-1217423 and the Israel Science Foundation grant no. 1147/09.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 411–424, 2013.

at most D. Local reconstruction algorithms provide very fast query access to a "corrected" version of G. That is, the local reconstruction algorithm should have in mind some $\widetilde{G}$ which has the property $\mathcal{P}_s$ and is also close to the original graph G. The goal of the local reconstruction algorithm is to provide very fast query access to the edges of $\widetilde{G}$ — that is, given a pair of vertices u, v in G, the algorithm should in sublinear time determine whether the edge (u, v) is in $\widetilde{G}$. We call such an algorithm a *local reconstructor* for $\mathcal{P}_s$. It can be useful to relax the condition that $\widetilde{G}$ has property $\mathcal{P}_s$ and only require that $\widetilde{G}$ has property $\mathcal{P}_{\phi(s)}$ which contains $\mathcal{P}_s$ but is possibly larger. For instance, if $\mathcal{P}_D$ is the property of having diameter at most D, we might only require $\widetilde{G}$ to have property $\mathcal{P}_{4D}$, i.e. having diameter at most $4D$.

In this paper we study local reconstruction algorithms for some of the most basic problems in graph theory, namely connectivity in undirected graphs, strong connectivity in directed graphs, k-connectivity in undirected graphs, and small diameter in undirected graphs. Such algorithms might be used to efficiently repair connectivity or small diameter in graphs. These are common issues that have been considered in various models in wireless networks and robotics (see for example [9], [25]).

Techniques for designing local reconstructors are often borrowed from property testing, and as noted by Brakerski [5] (as well as in this paper), reconstructors can be used to design property testers. Property testers have been studied extensively in the literature (see, for instance, [22], [10], [11] for some early works on the subject). A property tester for a property $\mathcal{P}$ takes as input a graph G and parameter ϵ, and accepts with high probability if G has $\mathcal{P}$ and rejects with high probability if G is ϵ-far from having $\mathcal{P}$. Similarly, a tolerant tester gets a graph G and parameters $\epsilon_1 \leq \epsilon_2$, and accepts with high probability if G is ϵ_1-close to having $\mathcal{P}$ and rejects with high probability if G is ϵ_2-far from having $\mathcal{P}$.

Our results. Our specific results are the following.

- Local reconstructors for the following properties: connectivity, strong connectivity, k-connectivity, having diameter at most D.
- A method of transforming our local reconstructor for k-connectivity (which provides query access to the adjacency *matrix* of the corrected graph) into an algorithm which provides query access to the adjacency *list* of the corrected graph (see Section 6 for the case $k = 1$). This is *not* a black-box transformation.
- We exploit a connection between local property reconstruction and property testing, observed by Brakerski [5] to obtain tolerant property testers for all of the above graph properties (property testing notions will be defined shortly).

Our approach. Our techniques are simple, yet seem to be quite powerful given their simplicity. For each of the above properties, the strategy for constructing a local reconstructor is the same. First, we designate a "super-node"; then elect "leader nodes" from which we add edges to the super-node. The main technique

we use to elect leaders is to initially independently assign a random rank to every node, and to declare a node a leader if it has the lowest rank among all nodes within a small neighborhood. This is more useful than simply choosing leaders at random, since sometimes we would like to guarantee we have a leader; for instance, for connectivity, we need at least one leader in each connected component if we want to *guarantee* that the corrected graph is connected.

Brakerski [5] gave a way to construct a tolerant tester from a local reconstructor and property tester. Since previous works ([11], [3], [15], [20]) give property testers for these properties we study, we obtain, as corollaries, that these properties have tolerant testers.

Related work. The notion of locally reconstructing a data set was introduced under the name *local filter* in [1] and further refined in [24]. In both works, the property of monotonicity of sequences was considered. A closely related work is [5], which introduces the concept of local reconstruction under the name *local restoring* and also shows a special case of our relationship between local reconstructors, property testers, and tolerant testers. Several other properties of graphs (expansion [13], bipartiteness [5], ρ-clique [5]), functions (Lipschitz [12], monotonicity [1,24,5]), geometric objects (convexity [8]) have been studied in the context of local reconstruction. The problem of testing the properties for which we give local reconstructors has been studied in multiple works—[11] for connectivity, [3] for strong connectivity, [11] and [15] for k-connectivity, and [20] for diameter. A tolerant tester for connectivity is also implied by the algorithm of [7] for approximating the number of connected components.

Organization. In Section 2, we formally define our model and the notions of local reconstructors, property testers, and tolerant testers, and in Section 3 we formally state our main results. In Section 4 and 5, we present our results for connectivity and strong connectivity. Section 6 serves as a brief interlude where we show how our local reconstructor for connectivity can be modified to give a neighbor oracle for the corrected graph G'. This procedure will then be used in Sections 7 and 8, in which we present our results for k-connectivity and small diameter respectively. Proofs for every claim can be found in the full version of this paper [6].

2 Preliminaries

We adopt the general sparse model of graphs as presented in [20], i.e. the graph is given as an adjacency list, and a query for a vertex v is either its degree, or an index i on which the i-th neighbor of v is returned (with respect to the representation of the neighbors as an ordered list). We assume there is some upper bound m on the number of edges of the graphs we work with, and distances are measured according to this, i.e. if k is the minimum number of edge deletions and insertions necessary to change one graph to the other, then their distance is k/m. We assume $m = \Omega(n)$ where n is the number of vertices in the graph.

Definition 1 ([20]). *The* distance between two graphs G_1, G_2, *denoted* $\mathrm{dist}(G_1, G_2)$, *is equal to the number of unordered pairs* (u, v) *such that* (u, v) *is an edge in one graph but not in the other, divided by* m. *A* property *is a subset of graphs. Throughout this paper we say that a graph has property* $\mathcal{P}$ *if it is contained in the subset* $\mathcal{P}$. *The* distance between a graph G and a property $\mathcal{P}$, *denoted* $\mathrm{dist}(G, \mathcal{P})$, *is equal to* $\mathrm{dist}(G, \mathcal{P}) = \min_{G' \in \mathcal{P}} \mathrm{dist}(G, G')$. *If* $\mathrm{dist}(G, \mathcal{P}) \leq \epsilon$, *then* G *is* ϵ-close to $\mathcal{P}$, *otherwise* G *is* ϵ-far from $\mathcal{P}$.

Parametrized properties. Our result relating local reconstructors to tolerant testers (Theorem 1) extends a result of Brakerski ([5]) to parametrized properties. A *parametrized property* $\mathcal{P}_s$ is a property belonging to a family $\{\mathcal{P}_s\}_s$ of properties parametrized by some parameter s. For example, the property $\mathcal{P}_D$ of having diameter at most D is a parametrized property, with the diameter D as the parameter.

2.1 Local Reconstructors

Definition 2. *For an undirected graph* $G = (V, E)$, *the* neighbor set of $v \in V$ *is the set* $N_G(v) = \{u \in V \mid (u, v) \in E\}$. *For a directed graph* $G = (V, E)$, *the* in-neighbor set and out-neighbor set of $v \in V$ *are respectively* $N_G^{\mathrm{in}}(v) = \{u \in V \mid (u, v) \in E\}$ *and* $N_G^{\mathrm{out}}(v) = \{u \in V \mid (v, u) \in E\}$

Definition 3 (Neighbor and edge oracles). *A* neighbor oracle *for a graph* G *is an algorithm which, given query* $v \in V$ *and either query* deg *or* i, *returns* $\deg(v)$ *or the* i*-th neighbor of* v *(with respect to some fixed ordering of the neighbor set) in* $O(1)$ *time. An* edge oracle for G *is an algorithm* E_G *which returns in* $O(1)$ *time* $E_G(u, v) = 1$ *if* $(u, v) \in E$ *and* $E_G(u, v) = 0$ *otherwise.*

Definition 4 (Local reconstructor). *Let* $\epsilon_1, \epsilon_2, \delta > 0$ *and let* $\phi : \mathbb{N} \to \mathbb{N}$. *An* $(\epsilon_1, \epsilon_2, \delta, \phi(\cdot))$-local reconstructor (LR) $\mathcal{R}$ *for a parametrized graph property* $\mathcal{P}_s$ *is a randomized algorithm with access to a neighbor oracle of a graph* G *that is* ϵ_1*-close to* $\mathcal{P}_s$, *which satisfies the following:*

- $\mathcal{R}$ *makes* $o(m)$ *queries to the neighbor oracle for* G *per query to* $\mathcal{R}$
- *There exists* $\widetilde{G} \in \mathcal{P}_{\phi(s)}$ *with* $\mathrm{dist}(G, \widetilde{G}) \leq \epsilon_2$ *such that* $\mathcal{R}$ *is an edge oracle for* $\widetilde{G}$, *with probability at least* $1 - \delta$ *(over the coin tosses of* $\mathcal{R}$*)*

An $(\epsilon_1, \epsilon_2, \delta)$-local reconstructor *for a non-parametrized graph property* $\mathcal{P}$ *is simply a* $(\epsilon_1, \epsilon_2, \delta, \phi(\cdot))$*-local reconstructor where* $\mathcal{P}$ *is viewed as the only property in its parametrized family and* ϕ *is the identity function. The* query complexity *of the local reconstructor is the number of queries* $\mathcal{R}$ *makes to the neighbor oracle for* G *on any query* (u, v). *We note that this definition differs from that of [12] because even if* $G \in \mathcal{P}$, *the reconstructed graph* $\widetilde{G}$ *may not equal* G *in general.*

2.2 Tolerant Testers

Tolerant testers (see [21]) are a generalization of property testers where the tester may accept if the input is close enough to having the property, where for property testers "close enough" means "distance zero".

Definition 5 (Tolerant tester). *Let* $\epsilon_1, \epsilon_2 > 0$ *and let* $\phi : \mathbb{N} \to \mathbb{N}$. *An* $(\epsilon_1, \epsilon_2, \phi(\cdot))$-tolerant tester $\mathcal{T}$ *for a parametrized graph property* $\mathcal{P}_s$ *is a randomized algorithm with query access to a neighbor oracle of an input graph* G *that satisfies the following:*

- $\mathcal{T}$ *makes* $o(m)$ *queries to the neighbor oracle for* G
- *If* G *is* ϵ_1*-close to* $\mathcal{P}_s$, *then* $\Pr[\mathcal{T} \text{ accepts }] \geq \frac{2}{3}$
- *If* G *is* ϵ_2*-far from* $\mathcal{P}_{\phi(s)}$, *then* $\Pr[\mathcal{T} \text{ accepts }] \leq \frac{1}{3}$

For a non-parametrized graph property $\mathcal{P}$, *an* (ϵ_1, ϵ_2)-tolerant *tester is defined similarly by viewing* $\mathcal{P}$ *as the single member of its parametrized family and taking* ϕ *to be the identity function.*

For a parametrized graph property, an $(\epsilon, \phi(\cdot))$-property tester *is simply a* $(0, \epsilon, \phi(\cdot))$*-tolerant tester, and for a non-parametrized graph property, an* ϵ-property tester *is defined analogously.*

3 Local Reconstructors and Tolerant Testers

We now show that our notion of local reconstructors can be used alongside property testers to construct tolerant testers for properties of sparse graphs. This idea is not new and can be found as [5, Theorem 3.1], but we extend the result for parametrized properties.

Theorem 1. *Let* $\mathcal{P}_s$ *be a parametrized property with an* $(\epsilon_1, \epsilon_2, \delta, \phi(\cdot))$*-local reconstructor* $\mathcal{R}$ *with query complexity* $q_{\mathcal{R}}$ *and suppose* $\mathcal{P}_{\phi(s)}$ *has a* $(\epsilon', \psi(\cdot))$*-property tester* $\mathcal{T}$ *with query complexity* $q_{\mathcal{T}}$. *Then for all* $\beta > 0$, $\mathcal{P}_s$ *has an* $(\epsilon_1, \epsilon_2 + \epsilon' + \beta, (\psi \circ \phi)(\cdot))$*-tolerant tester with query complexity* $O\left((1/\beta^2 + q_{\mathcal{T}})q_{\mathcal{R}}\right)$.

In this work, we give local reconstructors for several graph properties: connectivity in undirected graphs, strong connectivity in directed graphs, and small diameter in undirected graphs. To be precise, we prove the following in Sections 4, 5, 7 and 8 respectively.

Theorem 2. *There is an* $(\epsilon, (1+\alpha)\epsilon, \delta)$*-LR for connectivity with query complexity* $O\left(\frac{1}{\delta\alpha\epsilon}\right)$.

Theorem 3. *There is an* $(\epsilon, (4+\alpha)\epsilon, \delta)$*-LR for strong connectivity with query complexity* $O\left(\frac{1}{\delta\alpha\epsilon}\right)$.

Theorem 4. *There is an* $(\epsilon, (2+\alpha)\epsilon + \frac{ck}{2}, k(\delta+\gamma))$*-LR for* k*-connectivity with query complexity* $O\left(\left(\left(\frac{1}{c}+1\right)kt^3(t+k)\log(t+k)\log(Cn)\right)^k\right)$, *with* $C = \frac{1}{\ln(1/(1-\gamma))}$ *and* $t = \frac{\ln(Cn)}{\delta\alpha\epsilon}$.

Theorem 5. *There is an* $(\epsilon, (3+\alpha)\epsilon + \frac{1}{m} + c, \delta + \frac{1}{n}, \phi(s) = 2s+2)$*-LR for diameter at most* D *with query complexity* $O(\frac{1}{c\delta\alpha\epsilon}\Delta^{O(\Delta \log \Delta)} \log n)$ *where* $\Delta = (\overline{d}/\epsilon)^{O(1/\epsilon)}$ *and* $\overline{d} = 2m/n$ *is the bound on average degree.*

In light of Theorem 1, we have the following.

Corollary 1. *For all $\alpha, \beta, \epsilon > 0$, there is an $(\epsilon, (1+\alpha)\epsilon + \beta)$-tolerant tester for connectivity.*

Corollary 2. *For all $\alpha, \beta, \epsilon > 0$, there is an $(\epsilon, (4+\alpha)\epsilon + \beta)$-tolerant tester for strong connectivity.*

Corollary 3. *For all $\alpha, \beta, c, \epsilon > 0$, there is an $(\epsilon, (2k+\alpha)\epsilon + ck/2 + \beta)$-tolerant tester for k-connectivity.*

Corollary 4. *For all $D, \alpha, \beta, \epsilon > 0$ and constant $c < 1$, there is an $(\epsilon, (3+\alpha)\epsilon + \frac{1}{m} + c + \beta, 4D+6)$-tolerant tester for diameter at most D, for $n \geq \frac{k}{c}$.*

4 Local Reconstruction of Connectivity

High level description. The basic idea behind the algorithm is as follows. We designate a "super-node" v_0 and add edges from a few special vertices to v_0 so that the resulting graph is connected. Ideally, we have exactly one special vertex in each connected component, since this number of edges is both necessary and sufficient to make the graph connected. Therefore, we reduce the problem to defining a notion of "special" that can be determined quickly, that ensures that at least one node per component is special and that likely not too many extra nodes per component are special. How does a given vertex know whether it is special? Our algorithm tosses coins to randomly assign a rank $r(v) \in (0, 1]$ to each $v \in V(G)$. Then v can explore its connected component by performing a breadth-first search (BFS) and if v happens to have the lowest rank among all vertices encountered, then v is special. The only problem with this approach is that if v lies in a large connected component, then the algorithm makes too many queries to G to determine whether v is special. We fix this by limiting the BFS to K vertices, where K is a constant depending only on a few parameters, such as success probability and closeness. Components larger than size K do not contribute many more special vertices.

Choose K to be $\frac{m}{\delta\alpha\epsilon m - 1} = O\left(\frac{1}{\delta\alpha\epsilon}\right)$. The query complexity is $O(K) = O\left(\frac{1}{\delta\alpha\epsilon}\right)$. $\widetilde{G}$ is connected because, since the rank function attains a minimum on some vertex in each connected component, that vertex must get an edge to v_0. That $\widetilde{G}$ is close to G follows from the Markov's inequality and the fact that every component of size at most K contributes at most one edge and every remaining vertex contributes $1/K$ edges on average, and the fact that the number of components is bounded by $\epsilon m + 1$.

5 Local Reconstruction of Strong Connectivity

Query model. In our model, we assume our neighbor oracle has access to both the in-neighbor set and the out-neighbor set. This allows us to perform both backward and forward depth-first search (DFS), as well as undirected BFS, which is a BFS ignoring directions of edges.

High level description. The basic idea behind the algorithm is as follows. As in the undirected connectivity case, we designate a "super-node" v_0, but now we add arcs from a few special "transmitting" vertices to v_0 and also add arcs from v_0 to a few special "receiving" vertices. In order to make G strongly connected, we need to add at least one arc from the super-node to each source component and from each sink component to the super-node without adding too many extra arcs. A naïve approach is to emulate the strategy for connectivity: to decide if v is a transmitter, do a forward DFS from v and check if v has minimal rank (and analogously for receivers and backward DFS). Again, we can limit the search so that large components may have some extra special vertices. The problem with this approach is that a sink component could be extremely small (e.g. one vertex) with many vertices whose only outgoing arcs lead to the sink. In this case, all of these vertices would be special and receive an edge to v_0. Therefore we tweak our algorithm so that if v does a forward DFS and sees few vertices, then it checks if it is actually in a sink component. If so, then it is a transmitter; if not, then we do a limited *undirected BFS* from v and check minimality of rank. The procedure uses subroutines which respectively check if a vertex v is in a sink or a source of size less than K. We implement these subroutines using Tarjan's algorithm for finding strongly connected components, except stopping after only exploring all nodes reachable from v. We then show that if a directed graph is almost strongly connected, then it cannot have too many source, sink, or connected components, and therefore our algorithm likely does not add too many arcs.

6 Implementing a Neighbor Oracle with Connectivity Reconstructor

Our local reconstructors for k-connectivity and small diameter rely on the given graph G being connected. Even if G is not connected, it is close to being connected, and so one may hope to first make G into an intermediate connected graph G' using a local reconstructor for connectivity, and then run the local reconstructor for the desired property on G' to obtain $\widetilde{G}$. We would therefore like a neighbor oracle for G', but the local reconstructor only gives us an edge oracle for G'. We show how to modify the local reconstructor for connectivity to obtain a neighbor oracle for G', with only a slight loss in the parameters achieved.

As is, our connectivity reconstructor Connected from Section 4 is *almost* a neighbor oracle. Recall that the reconstructor works by selecting an arbitrary $v_0 \in G$ and adding edges from a few special vertices to v_0—no other edges are added. For a vertex $v \neq v_0$, $N_G(v) \subseteq N_{G'}(v) \subseteq N_G(v) \cup \{v_0\}$, thus $N_{G'}(v)$ can be computed in constant time. However, the problem arises when one queries $N_{G'}(v_0)$. Potentially $\Theta(n)$ edges are added to v_0 by the reconstructor, so computing $N_{G'}(v)$ queries Connected $O(n)$ times. We thus modify Connected to obtain the following.

Theorem 6. *Fix a positive constant $c < 1$. There is a randomized algorithm N, given access to the neighbor oracle N_G for G that is ϵ-close to being connected*

such that, with probability at least $1-\delta$*, there exists a connected graph* G' *that is* $((1+\alpha)\epsilon+c)$*-close to* G *and* N *is a neighbor oracle for* G'*, with query complexity* $O\left(\frac{1}{c\delta\alpha\epsilon}\right)$.

Proof. We modify CONNECTED as follows. Instead of designating one super-node v_0, we designate $c \cdot n$ super-nodes. Partition V into sets of size $1/c$ and assign each set in the partition to a distinct super-node. This can be implemented, for instance, by identifying $V = \{1, \ldots, n\}$, designating the super-nodes to be $\{1, \ldots, cn\}$, and for a given vertex $v \in V$, assign v to the super-node $h(v) = \lceil v/c \rceil$. For any v that would be connected to v_0, we instead connect it to $h(v)$. Additionally, we add the edges $(i, i+1)$ for all $i \in \{1, \ldots, cn-1\}$ to ensure connectivity. This adds a total of $cn-1$ edges, which constitute at most c-fraction of the edges. Call this modified local reconstructor MOD-CONNECTED. It is straightforward to see that MOD-CONNECTED has the same query complexity as CONNECTED. We now implement an algorithm to compute $N_{G'}$ as follows. Given a non-super-node v, its neighbor set could have grown by at most adding $h(v)$, so $N_{G'}(v)$ can be computed with $O(1)$ calls to MOD-CONNECTED. For a super node w, its neighbor set could have grown by at most $1/c+2$, since at most $1/c$ non-super-nodes could have been connected to w, and w is further connected to at most two super-nodes. Therefore $N_{G'}(w)$ can be computed with $O(1/c)$ calls to MOD-CONNECTED.

7 Local Reconstruction of k-Connectivity

7.1 Preliminaries

For a subset $U \subsetneq V$ of the vertices in a graph $G = (V, E)$, the *degree of* U, denoted $\deg(U)$, is equal to $\deg(U) = |\{(u, v) \in E \mid u \in U, v \in V \setminus U\}|$.

Definition 6 (k-connectivity). *An undirected graph* $G = (V, E)$ *is* k-connected *if for every* $U \subsetneq V$, $\deg(U) \geq k$.

An equivalent definition of k-connectivity, a result of Menger's theorem (see [16]), is that every pair of vertices has at least k edge-disjoint paths connecting them. An important notion of k-connectivity is that of an extreme set. Extreme sets are a generalization of connected components to the k-connectivity setting. Connected components are 0-extreme sets.

Definition 7. *A set* $U \subseteq V$ *is* ℓ*-extreme if* $\deg(U) = \ell$ *and* $\deg(W) > \ell$ *for every* $W \subsetneq U$.

It is straightforward from the definition that if a graph is $(k-1)$-connected and has no $(k-1)$-extreme sets, then it is in fact k-connected. Extreme sets satisfy some nice properties, which are used by [15] as well for property testing and distance approximation for k-connectivity. Two extreme sets are either disjoint or one is contained in the other (see [18]). If $W \subsetneq U$ and W is ℓ_W-extreme and U is ℓ_U-extreme, then $\ell_W > \ell_U$. Consequently, distinct ℓ-extreme sets are disjoint.

A graph G that is $(k-1)$-connected cannot have any ℓ-extreme sets for $\ell < k-1$. Moreover, the number of additional edges required to make G k-connected is at least half the number of $(k-1)$-extreme sets in G. This is simply because each $(k-1)$-extreme set requires at least one additional edge, and adding an edge to G meets the demand of at most two such sets.

7.2 High Level Description

The idea behind the algorithm is to simply iteratively make the graph j-connected, for $j = 1, 2, \ldots, k$. Let G_j be the corrected j-connected graph obtained from G. It suffices to use a neighbor oracle for G_{k-1} to implement a neighbor oracle for G_k. The base case $k = 1$ is addressed by Section 6.

Now suppose we have a neighbor oracle for $(k-1)$-connectivity and we wish to implement a neighbor oracle for k-connectivity. Again, we use a similar idea as in Sections 4 and 6. Specifically, we fix a positive constant $k/n \leq c < 1$ and designate a set $V_0 \subset V$ of $c \cdot n \geq k$ super-nodes, connecting them in a certain way to make the subgraph induced by V_0 k-connected (details in the next subsection). The idea is then to ensure at least one vertex from each extreme set contributes a new edge to a super-node. Again, we implement this by assigning all vertices a random rank independently and uniformly in $[0, 1)$ and searching a neighborhood of v up to t vertices, where t is appropriately chosen, and checking if it has minimal rank. Instead of doing this search via BFS, we use the extreme set search algorithm of [11]. The basic procedure satisfies the following: if v lies in a t-bounded extreme set, it finds this set with probability $\Theta(t^{-2})$, otherwise it never succeeds. We iterate the basic procedure a polylogarithmic number of times. If every iteration fails (which happens if v does not lie in a t-bounded extreme set) then we tell v to connect to a super-node with probability $\Theta\left(\frac{\log(n/t)}{t}\right)$. We then show that with high probability the resulting graph G_k is k-connected and that we do not add too many edges. This completes the edge oracle. We will also show how to implement these ideas carefully so that the edge oracle can be transformed into a neighbor oracle in order to make the recursion work.

7.3 Algorithm

Given our previous discussion, all that remains to implement the algorithm is to implement the following tasks:

- **Search:** Given $v \in V$ which lies in an extreme set S, find a neighborhood $U \subseteq S$ containing v such that $|U| \leq t$.
- **Decision:** Given $v \in V$, determine whether v should contribute an edge to V_0, and if so, to which $v' \in V_0$.

Implementing the Search Task. The goal of the search task is to detect that v lies in an extreme set of size at most t. We are now assuming the input graph is $(k-1)$-connected, so all extreme sets are $(k-1)$-extreme. The search task can be implemented by a method of [11] and [15] which runs in time $O(t^3 d \log(td))$

where d is a degree bound on the graph. Roughly, the procedure works by growing a set U' starting with $\{v\}$ and iteratively choosing a cut edge and adding the vertex on the other end of the edge into U'. The cut edge is chosen by assigning random weights to the edges, and choosing the edge with minimal weight. The procedure stops when the cut size is less than k or when $|U'| = t$.

Its running time is $O(td\log(td))$ but its success probability is only $\Omega(t^{-2})$ (that is, the probability that $U' = S$ when the procedure terminates), so the basic procedure is repeated $\Theta(t^2)$ times or until success. We can check if each run is successful by checking that the final set U' is a $(k-1)$-extreme set. This procedure is adapted in [20] to the general sparse model by noticing that no vertex of degree at least $t+k$ would ever be added to S, and hence the procedure has a running time of $O(t^3(t+k)\log(t+k))$. We actually want the probability that all vertices have a successful search to be at least $1-\gamma/2$, so we repeat the basic procedure $O(t^2\log(Cn))$ times for $C = \frac{1}{\ln(1/(1-\gamma/2))}$, yielding a time complexity of $O(t^3(t+k)\log(t+k)\log(Cn))$.

Implementing the Decision Task. For the decision task, it is helpful to first think about how to do it globally. First, we must hash each $v \in V$ to a set $h(v)$ of k super-nodes in V_0. This can be implemented as follows. Label $V = \{1,\ldots,n\}$ and $V_0 = \{1,\ldots,cn\}$, and define $h(v) = \{\lceil v/c\rceil, \lceil v/c\rceil + 1, \ldots, \lceil v/c\rceil + (k-1)\}$ where the entries are taken modulo cn. The specific way we do this hashing is not important—it suffices to guarantee the following properties: (1) for every $v \in V$, $|h(v)| \geq k$ and (2) for every super-node $v' \in V_0$, there are at most a constant number, independent of n, of v such that $v' \in h(v)$, and that these v are easily computable given v'; our method guarantees the constant $\frac{k}{c}$, which is the best one can hope for given Property 1. Property 1 will be used later to ensure that the resulting graph is k-connected (Lemma 2). Property 2 ensures, by the same reasoning as in Section 6, that computing N_{G_k} makes at most $O\left(\frac{k}{c}+k\right)$ calls to E_{G_k}.

Now, to decide whether v should be connected to a super-node, do an extreme set search to find a neighborhood U, of size at most t, containing v, and check if v has minimal rank in U, where the rank is the randomly assigned rank given in the high level description. If so, then mark v as *successful*. If the extreme set search fails, i.e. all $\Theta(t^2\log n)$ iterations fail, then mark v as successful with probability $\frac{\ln(Cn/t)}{t}$ where $C = \frac{1}{\ln(1/(1-\gamma/2))}$ again, which can be implemented by checking if the rank of v is less than $\frac{\ln(Cn/t)}{t}$. If v is successful, then find the lexicographically smallest super-node $v' \in h(v)$ to which v is not already connected. If none exist, then do nothing; otherwise, connect v to v'.

We also want the subgraph induced by V_0 to be k-connected to help ensure that G_k is k-connected (Lemma 2). Globally, from each $i \in V_0$ we add an edge to $i+1, i+2, \ldots, i+\lceil k/2\rceil$, taken modulo cn. This ensures that the subgraph induced by V_0 is k-connected (Lemma 1). Locally, this is implemented as follows. If $(i,j) \in [cn]^2$ is queried, if the edge is already in G_{k-1}, then the local reconstructor returns 1, otherwise it returns 1 if and only if $j \in \{i+1, i+2, \ldots, i+\lceil k/2\rceil \pmod{cn}\}$ or $i \in \{j+1, j+2, \ldots, j+\lceil k/2\rceil \pmod{cn}\}$.

Lemma 1. *V_0 is k-connected.*

Lemma 2. *With probability at least $1-\gamma$, G_k is k-connected.*

Lemma 3. *With probability at least $1-\delta$, the number of edges added is at most*

$$2\epsilon m + ckn/2 + \frac{n\ln(Cn/t)}{\delta t}.$$

Proof (Proof of Theorem 4). Set $C = \frac{1}{\ln(1/(1-\gamma))}$ and $t = \frac{\ln(Cn)}{\delta\alpha\epsilon}$. By Lemma 2 our resulting graph is k-connected with probability at least $1-\gamma$ and by Lemma 3, $\mathrm{dist}(G_{k-1}, G_k) \leq (2+\alpha)\epsilon + ck/2$ with probability $1-\delta$ and therefore by induction $\mathrm{dist}(G, G_k) \leq (2+\alpha)k\epsilon + ck^2/2$. This can be improved to $(2+\alpha)k\epsilon + ck/2$ by noting that the same super-nodes and the same edges between them can be used for all intermediate graphs $G_1, \ldots, G_{k-1}$. The success probability is at least $(1-\delta)^k(1-\gamma)^k \geq 1-k(\delta+\gamma)$. The query complexity for correcting G_{k-1} to G_k is $O(t^3(t+k)\log(t+k)\log(Cn))$ queries to $N_{G_{k-1}}$. But each call to $N_{G_{k-1}}$ takes $O\left(\left(\frac{1}{c}+1\right)k\right)$ calls to $E_{G_{k-1}}$. By induction, the query complexity of E_{G_k} is $O\left(\left(\left(\frac{1}{c}+1\right)k\right)^k \cdot t^{3k}(t+k)^k \log^k(t+k)\log^k(Cn)\right)$.

8 Local Reconstruction of Small Diameter

8.1 Preliminaries

Definition 8. *Let G be a graph with adjacency matrix A. For an integer k, let G^k be the graph on the same vertex set as G, with adjacency matrix A^k (boolean arithmetic).*

Proposition 1. *Let G be a graph and let $D > 0$ be an integer. Then* $\mathrm{diam}\, G \leq D$ *if and only if G^D is a complete graph.*

8.2 High Level Description

The basic idea behind the algorithm is as follows. Again, we designate a "super-node" v_0 and add edges between v_0 and a few special vertices. If the input graph is close to having diameter at most D, then we aim for our reconstructed graph to have diameter at most $2D+2$. We show that if G is close to having diameter at most D, then we have an upper bound on the size of any independent set in G^D (Lemma 4 and Corollary 5).

Ideally, then, we want our special vertices (those that get an edge to v_0) to be a dominating set in G^D that is not too large. If we add edges from the dominating set in G^D to v_0, then our resulting graph $\widetilde{G}$ has diameter at most $2D+2$, since any vertex can reach some vertex in the dominating set within D steps, and hence v_0 within $D+1$ steps. If this dominating set is a maximal independent set, then our upper bound on the size of independent sets in G^D also upper bounds the number of edges we add. However, all known algorithms for locally

computing a maximal independent set have query complexity bounded in terms of the maximum degree of the graph. A variant of the algorithm found in [17] has been analyzed by [26] and [19] to run in expected time bounded in terms of the average degree of the graph, but this average is taken over not only coin tosses of the algorithm but also all possible queries. This is undesirable for us since we want a uniform bound on the query complexity for any potential query vertex, given a "good" set of coin tosses. We want an algorithm such that, for most sets of coin tosses, we get the correct answer *everywhere*, whereas the algorithm of [17] leaves open the possibility of failure for some queries, regardless of the coin tosses. If some queries give the wrong answer, then the fact that our reconstructed graph retains the property is compromised. Instead, we do something less optimal but good enough. Note that adding edges does not increase diameter. Instead of using a maximal independent set, we settle for a dominating set in G^D as long as we can still control its size.

8.3 Properties of Graphs Close to Having Small Diameter

The following lemma and subsequent corollary state that if a graph G is close to having diameter at most D, then no independent set in G^D can be very large.

Lemma 4. *Let $v_1, \ldots, v_k$ be an independent set in G^D and let H be the subgraph of G^D induced by this set. Let (s,t) be an edge not in $E(G)$ and let $G' = (V(G), E(G) \cup \{(s,t)\})$ be the graph obtained by adding (s,t) to G. Let H' be the subgraph of $(G')^D$ induced by $v_1, \ldots, v_k$. Then, for some $i \in \{1, \ldots, k\}$, all edges in H' (if any) are incident to v_i. In particular, if G^D has an independent set of size k, then $(G')^D$ has an independent set of size $k-1$.*

Corollary 5. *Suppose G is ϵ-close to having diameter $\leq D$. Then any independent set in G^D has size at most $\epsilon m + 1$.*

Finally, we will use a result from [2], [20] which we will restate in our own terms below (the theorem number we reference is from [2]):

Theorem 7 ([2, Theorem 3.1]). *Any connected graph G is $(\frac{2n}{Dm})$-close to having diameter $\leq D$.*

8.4 Algorithm

We start by fixing a super-node $v_0 \in G$. Given our discussion in the high level description, it remains to implement the selection of a small dominating set. To this end, we create a dominating set S by first adding the set H of high-degree vertices into S and then using the local maximal independent set algorithm found in [23], which is based on Luby's algorithm ([14]), on G^D with H and its vertices' neighbors (in G^D) removed to create an independent set M. Then let $S = H \cup M$.

By high-degree vertex we mean a vertex with degree greater than $\overline{d}/\epsilon$, of which there are at most ϵn, where $\overline{d} = \frac{2m}{n}$ is a bound on the average degree.

We will also consider the super-node v_0 a high-degree vertex for our purposes. To implement locally choosing a maximal independent set, we define a subroutine $\text{MIS}_D(v)$ as follows. On input v, simulate the local maximal independent set described in [23], except explore all neighbors within D steps from v rather than just immediate neighbors, and automatically reject (leave out of the MIS) if the algorithm encounters any vertex with degree exceeding $\overline{d}/\epsilon$. Note that the running time of the local MIS algorithm given in [23] is bounded in terms of the maximum degree of the graph. This is not problematic since our variant of the algorithm ignores any vertex with degree greater than $\overline{d}/\epsilon$, hence the effective maximum degree of our graph G for the purposes of the algorithm is $\overline{d}/\epsilon$, so the effective maximum degree of G^D is $(\overline{d}/\epsilon)^D$.

One final challenge is that if D is large, say $\Theta(\log n)$, then our query complexity bound in terms of our effective degree is no longer sublinear. We work around this by using Theorem 7, which states that every connected graph is ϵ-close to having diameter at most $\frac{2n}{\epsilon m} = O(1/\epsilon)$. Therefore, we can aim for achieving diameter $K = \min\{D, \frac{2n}{\epsilon m}\}$ so that our effective degree is $(\overline{d}/\epsilon)^K = (\overline{d}/\epsilon)^{O(1/\epsilon)}$. Of course, this only works if our graph is connected to begin with. Therefore, we use the neighbor oracle for the connected correction G' of G, given in Section 6. The idea is then to first make G into a connected graph G', and then reconstruct a small diameter graph $\widetilde{G}$ out of G'.

References

1. Ailon, N., Chazelle, B., Comandur, S., Liu, D.: Property-preserving data reconstruction. In: Proc. 15th International Symposium on Algorithms and Computation, pp. 16–27 (2004)
2. Alon, N., Gyàrfàs, A., Ruszinkò, M.: Decreasing the diameter of bounded degree graphs. J. Graph Theory 35, 161–172 (2000)
3. Bender, M.A., Ron, D.: Testing properties of directed graphs: acyclicity and connectivity. Random Structures and Algorithms 20, 184–205 (2002)
4. Bhattacharyya, A., Grigorescu, E., Jha, M., Jung, K., Raskhodnikova, S., Woodruff, D.P.: Lower bounds for local monotonicity reconstruction from transitive-closure spanners. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, pp. 448–461. Springer, Heidelberg (2010)
5. Brakerski, Z.: Local property restoring. Manuscript (2008)
6. Campagna, A., Guo, A., Rubinfeld, R.: Local reconstructors and tolerant testers for connectivity and diameter. ArXiv preprint. arXiv.DS:1208.2956
7. Chazelle, B., Rubinfeld, R., Trevisan, L.: Approximating the Minimum Spanning Tree Weight in Sublinear Time. SIAM J. Comput. 34(6), 1370–1379 (2005)
8. Chazelle, B., Seshadhri, C.: Online geometric reconstruction. In: Proc. 22nd ACM Symposium on Computational Geometry, pp. 386–394 (2006)
9. Derbakova, A., Correll, N., Rus, D.: Decentralized self-repair to maintain connectivity and coverage in networked multi-robot systems. In: Proc. IEEE International Conference on Robotics and Automation (2011)
10. Goldreich, O., Goldwasser, S., Ron, D.: Property Testing and its Connection to Learning and Approximation. J. ACM 45(4), 653–750 (1998)

11. Goldreich, O., Ron, D.: Property testing in bounded degree graphs. Algorithmica 32, 302–343 (2002)
12. Jha, M., Raskhodnikova, S.: Testing and reconstruction of Lipschitz functions with applications to data privacy. In: Proc. 52nd Annual IEEE Symposium on Foundations of Computer Science, pp. 433–442
13. Kale, S., Peres, Y., Seshadhri, C.: Noise tolerance of expanders and sublinear expander reconstruction. In: Proc. 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 719–728 (2008)
14. Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM Journal on Computing 15(4), 1036–1053 (1986)
15. Marko, S., Ron, D.: Approximating the distance to properties in bounded-degree and general sparse graphs. ACM Transactions on Algorithms 5(2) (2009)
16. Menger, K.: Zur allgemeinen Kurventheorie. Fund. Math. 10, 96–115 (1927)
17. Nguyen, H.N., Onak, K.: Constant-time approximation algorithms via local improvements. In: Proc. 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 327–336 (2008)
18. Noar, D., Gusfield, D., Martel, C.: A fast algorithm for optimally increasing the edge connectivity. SICOMP 26(4), 1139–1165 (1997)
19. Onak, K., Ron, D., Rosen, M., Rubinfeld, R.: A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In: Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1123–1131 (2012)
20. Parnas, M., Ron, D.: Testing the diameter of graphs. Random Structures and Algorithms 20, 165–183 (2002)
21. Parnas, M., Ron, D., Rubinfeld, R.: Tolerant property testing and distance approximation. Journal of Computer and System Sciences 72, 1012–1042 (2006)
22. Rubinfeld, R., Sudan, M.: Robust Characterizations of Polynomials with Applications to Program Testing. SIAM J. Comput. 25(2), 252–271 (1996)
23. Rubinfeld, R., Tamir, G., Vardi, S., Xie, N.: Fast local computation algorithms. In: Proc. 2nd Symposium on Innovations in Computer Science, pp. 223–238 (2011)
24. Saks, M.E., Seshadhri, C.: Local monotonicity reconstruction. SIAM Journal on Computing 39, 2897–2926 (2010)
25. Stump, E., Jadbabaie, A., Kumar, V.: Connectivity management in mobile robot teams. In: Proc. IEEE International Conference on Robotics and Automation (2008)
26. Yoshida, Y., Yamamoto, M., Ito, H.: An improved constant-time approximation algorithm for maximum matchings. In: Proc. 41st ACM Symposium on Theory of Computing, pp. 225–234 (2009)

An Optimal Lower Bound for Monotonicity Testing over Hypergrids

Deeparnab Chakrabarty[1] and C. Seshadhri[2,⋆]

[1] Microsoft Research India
dechakr@microsoft.com
[2] Sandia National Labs, Livermore
scomand@sandia.gov

Abstract. For positive integers n, d, consider the hypergrid $[n]^d$ with the coordinate-wise product partial ordering denoted by $\prec$. A function $f : [n]^d \to \mathbb{N}$ is monotone if $\forall x \prec y$, $f(x) \leq f(y)$. A function f is ε-far from monotone if at least an ε-fraction of values must be changed to make f monotone. Given a parameter ε, a *monotonicity tester* must distinguish with high probability a monotone function from one that is ε-far.

We prove that any (adaptive, two-sided) monotonicity tester for functions $f : [n]^d \to \mathbb{N}$ must make $\Omega(\varepsilon^{-1} d \log n - \varepsilon^{-1} \log \varepsilon^{-1})$ queries. Recent upper bounds show the existence of $O(\varepsilon^{-1} d \log n)$ query monotonicity testers for hypergrids. This closes the question of monotonicity testing for hypergrids over arbitrary ranges. The previous best lower bound for general hypergrids was a non-adaptive bound of $\Omega(d \log n)$.

Keywords: Monotonicity testing, sublinear algorithms, lower bounds.

1 Introduction

Given query access to a function $f : \mathbf{D} \to \mathbf{R}$, the field of *property testing* [1,2] deals with the problem of determining properties of f without reading all of it. Monotonicity testing [3] is a classic problem in property testing. Consider a function $f : \mathbf{D} \to \mathbf{R}$, where $\mathbf{D}$ is some partial order given by "$\prec$", and $\mathbf{R}$ is a total order. The function f is monotone if for all $x \prec y$ (in $\mathbf{D}$), $f(x) \leq f(y)$. The *distance to monotonicity* of f is the minimum fraction of values that need to be modified to make f monotone. More precisely, define the distance between functions $d(f, g)$ as $|\{x : f(x) \neq g(x)\}|/|\mathbf{D}|$. Let $\mathcal{M}$ be the set of all monotone functions. Then the distance to monotonicity of f is $\min_{g \in \mathcal{M}} d(f, g)$.

A function is called ε-far from monotone if the distance to monotonicity is at least ε. A *property tester for monotonicity* is a, possibly randomized, algorithm that takes as input a distance parameter $\varepsilon \in (0, 1)$, error parameter $\delta \in [0, 1]$,

⋆ Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 425–435, 2013.

and query access to an arbitrary f. If f is monotone, then the tester must accept with probability $> 1-\delta$. If it is ε-far from monotone, then the tester rejects with probability $> 1-\delta$. (If neither, then the tester is allowed to do anything.) The aim is to design a property tester using as few queries as possible. A tester is called *one-sided* if it always accepts a monotone function. A tester is called *non-adaptive* if the queries made do not depend on the function values. The most general tester is an adaptive two-sided tester.

Monotonicity testing has a rich history and the hypergrid domain, $[n]^d$, has received special attention. The boolean hypercube ($n = 2$) and the total order ($d = 1$) are special instances of hypergrids. Following a long line of work [4,3,5,6,7,8,9,10,11,12,13,14], previous work of the authors [15] shows the existence of $O(\varepsilon^{-1} d \log n)$-query monotonicity testers. Our result is a matching adaptive lower bound that is optimal in all parameters (for unbounded range functions). This closes the question of monotonicity testing for unbounded ranges on hypergrids. This is also the first adaptive bound for monotonicity testing on general hypergrids.

Theorem 1. *Any (adaptive, two-sided) monotonicity tester for functions $f : [n]^d \to \mathbb{N}$ requires $\Omega(\varepsilon^{-1} d \log n - \varepsilon^{-1} \log \varepsilon^{-1})$ queries.*

1.1 Previous Work

The problem of monotonicity testing was introduced by Goldreich et al. [3], with an $O(n/\varepsilon)$ tester for functions $f : \{0,1\}^n \to \{0,1\}$. The first tester for general hypergrids was given by Dodis et al. [5]. The upper bound of $O(\varepsilon^{-1} d \log n)$ for monotonicity testing was recently proven in [15]. We refer the interested reader to the introduction of [15] for a more detailed history of previous upper bounds.

There have been numerous lower bounds for monotonicity testing. We begin by summarizing the state of the art. The known adaptive lower bounds are $\Omega(\log n)$ for the total order $[n]$ by Fischer [9], and $\Omega(d/\varepsilon)$ for the boolean hypercube $\{0,1\}^d$ by Brody [16]. For general hypergrids, Blais, Raskhodnikova, and Yaroslavtsev [17] recently proved the first result, a non-adaptive lower bound of $\Omega(d \log n)$. Theorem 1 is the first adaptive bound for monotonicity testing on hypergrids and is optimal (for arbitrary ranges) in all parameters.

Now for the chronological documentation. The first lower bound was the non-adaptive bound of $\Omega(\log n)$ for the total order $[n]$ by Ergun et al. [4]. This was extended by Fischer [9] to an (optimal) adaptive bound. For the hypercube domain $\{0,1\}^d$, Fischer et al. [7] proved the first non-adaptive lower bound of $\Omega(\sqrt{d})$. (This was proven even for the range $\{0,1\}$.) This was improved to $\Omega(d/\varepsilon)$ by Brïet et al. [18]. Blais, Brody, and Matulef [14] gave an ingenious reduction from communication complexity to prove an adaptive, two-sided bound of $\Omega(d)$. (Honing this reduction, Brody [16] improved this bound to $\Omega(d/\varepsilon)$.) The non-adaptive lower bounds of Blais, Raskhodnikova, and Yaroslavtsev [17] were also achieved through communication complexity reductions.

We note that our theorem only holds when the range is $\mathbb{N}$, while some previous results hold for restricted ranges. The results of [14,16] provide lower bounds for

range $[\sqrt{d}]$. The non-adaptive bound of [17] holds even when the range is $[nd]$. In that sense, the communication complexity reductions provide stronger lower bounds than our result.

1.2 Main Ideas

The starting point of this work is the result of Fischer [9], an adaptive lower bound for monotonicity testing for functions $f : [n] \to \mathbb{N}$. He shows that adaptive testers can be converted to comparison-based testers, using Ramsey theory arguments. A comparison-based tester for $[n]$ can be easily converted to a non-adaptive tester, for which an $\Omega(\log n)$ bound was previously known. We make a fairly simple observation. The main part of Fischer's proof actually goes through for functions over *any partial order*, so it suffices to prove lower bounds for comparison-based testers. (The reduction to non-adaptive testers only holds for $[n]$.)

We then prove a comparison-based lower bound of $\Omega(\varepsilon^{-1} d \log n - \varepsilon^{-1} \log \varepsilon^{-1})$ for the domain $[n]^d$. As usual, Yao's minimax lemma allows us to focus on determinstic lower bounds over some distribution of functions. The major challenge in proving (even non-adaptive) lower bounds for monotonicity is that the tester might make decisions based on the actual values that it sees. Great care is required to construct a distribution over functions whose monotonicity status cannot be decided by simply looking at the values. But a comparison-based tester has no such power, and optimal lower bounds over all parameters can be obtained with a fairly clean distribution.

2 The Reduction to Comparison Based Testers

Consider the family of functions $f : \mathbf{D} \to \mathbf{R}$, where $\mathbf{D}$ is some partial order, and $\mathbf{R} \subseteq \mathbb{N}$. We will assume that f always takes distinct values, so $\forall x, y, f(x) \neq f(y)$. Since we are proving lower bounds, this is no loss of generality.

Definition 1. *An algorithm $\mathcal{A}$ is a* (t, ε, δ)-monotonicity tester *if $\mathcal{A}$ has the following properties. For any $f : \mathbf{D} \to \mathbf{R}$, the algorithm $\mathcal{A}$ makes t (possibly randomized) queries to f and then outputs either "accept" or "reject". If f is monotone, then $\mathcal{A}$ accepts with probability $> 1 - \delta$. If f is ε-far from monotone, then $\mathcal{A}$ rejects with probability $> 1 - \delta$.*

Given a positive integer s, let $\mathbf{D}^s$ denote the collection of *ordered*, s-tupled vectors with each entry in $\mathbf{D}$. We define two symbols `acc` and `rej`, and denote $\mathbf{D}' = \mathbf{D} \cup \{\texttt{acc}, \texttt{rej}\}$. Any (t, ε, δ)-tester can be completely specified by the following family of functions. For all $s \leq t$, $\mathbf{x} \in \mathbf{D}^s$, $y \in \mathbf{D}'$, we consider a function $p^y_{\mathbf{x}} : \mathbf{R}^s \to [0, 1]$, with the semantic that for any $\mathbf{a} \in \mathbf{R}^s$, $p^y_{\mathbf{x}}(\mathbf{a})$ denotes the probability the tester queries y as the $(s+1)$th query, given that the first s queries are $\mathbf{x}_1, \ldots, \mathbf{x}_s$ and $f(\mathbf{x}_i) = \mathbf{a}_i$ for $1 \leq i \leq s$. By querying `acc`, `rej` we imply returning accept or reject. These functions satisfy the following properties.

$$\forall s \leq t,\ \forall \mathbf{x} \in \mathbf{D}^s,\ \forall \mathbf{a} \in \mathbf{R}^s,\ \sum_{y \in \mathbf{D}'} p^y_{\mathbf{x}}(\mathbf{a}) = 1 \tag{1}$$

$$\forall \mathbf{x} \in \mathbf{D}^t,\ \forall y \in \mathbf{D},\ \forall \mathbf{a} \in \mathbf{R}^t,\ p^y_{\mathbf{x}}(\mathbf{a}) = 0 \tag{2}$$

Eq. (1) ensures the decisions of the tester at step $(s+1)$ must form a probability distribution. Eq. (2) implies that the tester makes at most t queries.

For any positive integer s, let $\mathbf{R}^{(s)}$ denote the set of *unordered* subets of $\mathbf{R}$ of cardinality s. For reasons that will soon become clear, we introduce new functions as follows. For each s, $\mathbf{x} \in \mathbf{D}^s$, $y \in \mathbf{D}'$, and *each permutation* $\sigma : [s] \to [s]$, we associate functions $q^y_{\mathbf{x},\sigma} : \mathbf{R}^{(s)} \to [0,1]$, with the semantic

$$\text{For any set } S = (a_1 < a_2 < \cdots < a_s) \in \mathbf{R}^{(s)}, \qquad q^y_{\mathbf{x},\sigma}(S) := p^y_{\mathbf{x}}(a_{\sigma(1)}, \ldots, a_{\sigma(s)})$$

That is, $q^y_{\mathbf{x},\sigma_s}(S)$ sorts the answers in S in increasing order, permutes them according to σ, and passes the permuted ordered tuple to $p^y_{\mathbf{x}}$. Any adaptive tester can be specified by these functions. The important point to note is that they are finitely many such functions; their number is upper bounded by $(t|\mathbf{D}|)^{t+1}$. These q-functions allow us to define comparison based testers.

Definition 2. *A monotonicity tester $\mathcal{A}$ is* comparison-based *if for all s,$\mathbf{x} \in \mathbf{D}^s, y \in \mathbf{D}'$, and permutations $\sigma : [s] \to [s]$, the function $q^y_{\mathbf{x},\sigma}$ is a constant function on $\mathbf{R}^{(s)}$. In other words, the $(s+1)$th decision of the tester given that the first s questions is $\mathbf{x}$, depends only on the* ordering *of the answers received, and not on the values of the answers.*

The following theorem is implicit in the work of Fischer [9].

Theorem 2. *Suppose there exists a (t, ε, δ)-monotonicity tester for functions $f : \mathbf{D} \to \mathbb{N}$. Then there exists a comparison-based $(t, \varepsilon, 2\delta)$-monotonicity tester for functions $f : \mathbf{D} \to \mathbb{N}$.*

This implies that a comparison-based lower bound suffices for proving a general lower bound on monotonicity testing. We provide a proof of the above theorem in the next section for completeness.

2.1 Performing the Reduction

We basically present Fischer's argument, observing that $\mathbf{D}$ can be any partial order. A monotonicity tester is called *discrete* if the corresponding functions $p^y_{\mathbf{x}}$ can only take values in $\{i/K \ :\ 0 \leq i \leq K\}$ for some finite K. Note that this implies the functions $q^y_{\mathbf{x},\sigma}$ also take discrete values.

Lemma 1. *Suppose there exists a (t, ε, δ)-monotonicity tester $\mathcal{A}$ for functions $f : \mathbf{D} \to \mathbb{N}$. Then there exists a discrete $(t, \varepsilon, 2\delta)$-monotonicity tester for these functions.*

Proof. We do a rounding on the p-functions. Let $K = 100t|\mathbf{D}|^t/\delta^2$. Start with the p-functions of the (t, ε, δ)-tester $\mathcal{A}$. For $y \in \mathbf{D} \cup \mathtt{acc}$, $\mathbf{x} \in \mathbf{D}^s$, $\mathbf{a} \in \mathbf{R}^s$, let $\hat{p}^y_{\mathbf{x}}(\mathbf{a})$ be the largest value in $\{i/K \mid 0 \le i \le K\}$ at most $p^y_{\mathbf{x}}(\mathbf{a})$. Set $\hat{p}^{\mathtt{rej}}_{\mathbf{x}}(\mathbf{a})$ so that Eq. (1) is maintained.

Note that for $y \in \mathbf{D} \cup \mathtt{acc}$, if $p^y_{\mathbf{x}}(\mathbf{a}) > 10t/(\delta K)$, then

$$\left(1 - \frac{\delta}{10t}\right) p^y_{\mathbf{x}}(\mathbf{a}) \;\le\; \hat{p}^y_{\mathbf{x}}(\mathbf{a}) \;\le\; p^y_{\mathbf{x}}(\mathbf{a}).$$

Furthermore, $\hat{p}^{\mathtt{rej}}_{\mathbf{x}}(\mathbf{a}) \ge p^{\mathtt{rej}}_{\mathbf{x}}(\mathbf{a})$.

The $\hat{p}$-functions describe a new discrete tester $\mathcal{A}'$ that makes at most t queries. We argue that $\mathcal{A}'$ is a $(t, \varepsilon, 2\delta)$-tester. Given a function f that is either monotone or ε-far from monotone, consider a sequence of queries $x_1, \ldots, x_s$ after which $\mathcal{A}$ returns a *correct* decision $\aleph$. Call such a sequence good, and let α denote the probability this occurs. We know that the sum of probabilities over all good query sequences is at least $(1 - \delta)$. Now,

$$\alpha := p^{x_1} \cdot\; p^{x_2}_{x_1}(f(x_1)) \cdot\; p^{x_3}_{(x_1,x_2)}(f(x_1), f(x_2)) \cdots\; p^{\aleph}_{(x_1,\ldots,x_s)}(f(x_1), \ldots, f(x_s))$$

Two cases arise. Suppose all of the probabilities in the RHS are $\ge 10t/\delta K$. Then, the probability of this good sequence arising in $\mathcal{A}'$ is at least $(1-\delta/10t)^t\alpha \ge \alpha(1-\delta/2)$. Otherwise, suppose some probability in the RHS is $< 10t/\delta K$. Then the total probability mass on such good sequences in $\mathcal{A}$ is at most $10t/\delta K \cdot |\mathbf{D}|^t \le \delta/2$. Therefore, the probability of good sequences in $\mathcal{A}'$ is at least $(1-3\delta/2)(1-\delta/2) \ge 1 - 2\delta$. That is, $\mathcal{A}'$ is a $(t, \varepsilon, 2\delta)$ tester.

We introduce some Ramsey theory terminology. For any positive integer i, a *finite* coloring of $\mathbb{N}^{(i)}$ is a function $\mathtt{col}_i : \mathbb{N}^{(i)} \to \{1, \ldots, C\}$ for some finite number C. An infinite set $X \subseteq \mathbb{N}$ is called *monochromatic* w.r.t $\mathtt{col}_i$ if for all sets $A, B \in X^{(i)}$, $\mathtt{col}_i(A) = \mathtt{col}_i(B)$. A *$k$-wise* finite coloring of $\mathbb{N}$ is a collection of k colorings $\mathtt{col}_1, \ldots, \mathtt{col}_k$. (Note that each coloring is over different sized tuples.) An infinite set $X \subseteq \mathbb{N}$ is k-wise monochromatic if X is monochromatic w.r.t. all the $\mathtt{col}_i$s.

The following is a simple variant of Ramsey's original theorem. (We closely follow the proof of Ramsey's theorem as given in Chap VI, Theorem 4 of [19].)

Theorem 3. *For any k-wise finite coloring of $\mathbb{N}$, there is an infinite k-wise monochromatic set $X \subseteq \mathbb{N}$.*

Proof. We proceed by induction on k. If $k = 1$, then this is trivially true; let X be the maximum color class. Since the coloring is finite, X is infinite. We will now iteratively construct an infinite set of $\mathbb{N}$ via induction.

Start with a_0 being the minimum element in $\mathbb{N}$. Consider a $(k-1)$-wise coloring of $(\mathbb{N}\backslash\{a_0\})$ $\mathtt{col}'_1, \ldots, \mathtt{col}'_{k-1}$, where $\mathtt{col}'_i(S) := \mathtt{col}_{i+1}(S \cup a_0)$. By the induction hypothesis, there exists an infinite $(k-1)$-wise monochromatic set $A_0 \subseteq \mathbb{N}\backslash\{a_0\}$ with respect to coloring $\mathtt{col}'_i$s. That is, for $1 \le i \le k$, and any set $S, T \subseteq A_0$ with $|S| = |T| = i - 1$, we have $\mathtt{col}_i(a_0 \cup S) = \mathtt{col}_i(a_0 \cup T) = C^0_i$, say. Denote the collection of these colors as a vector $\mathbf{C}_0 = (C^0_1, C^0_2, \ldots, C^0_k)$.

Subsequently, let a_1 be the minimum element in A_0, and consider the $(k-1)$-wise coloring $\texttt{col}'$ of $(A_0 \setminus \{a_1\})$ where $\texttt{col}'_i(S) = \texttt{col}_{i+1}(S \cup \{a_1\})$ for $S \subseteq A_0 \setminus \{a_1\}$. Again, the induction hypothesis yields an infinite $(k-1)$-wise monochromatic set A_1 as before, and similarly the vector $\mathbf{C}_1$. Continuing this procedure, we get an infinite sequence $a_0, a_1, a_2, \ldots$ of natural numbers, an infinite sequence of vectors of k colors $\mathbf{C}_0, \mathbf{C}_1, \ldots$, and an infinite nested sequence of infinite sets $A_0 \supset A_1 \supset A_2 \ldots$. Every A_r contains $a_s, \forall s > r$ and by construction, any set $(\{a_r\} \cup S)$, $S \subseteq A_r$, $|S| = i-1$, has color C_r^i. Since there are only finitely many colors, some vector of colors occurs infinitely often as $\mathbf{C}_{r_1}, \mathbf{C}_{r_2}, \ldots$. The corresponding infinite sequence of elements $a_{r_1}, a_{r_2}, \ldots$ is k-wise monochromatic.

Proof. (of Theorem 2) Suppose there exists a (t, ε, δ)-tester for functions $f : \mathbf{D} \to \mathbb{N}$. We need to show there is a comparison-based $(t, \varepsilon, 2\delta)$-tester for such functions.

By Lemma 1, there is a discrete $(t, \varepsilon, 2\delta)$-tester $\mathcal{A}$. Equivalently, we have the functions $q^y_{\mathbf{x},\sigma}$ as described in the previous section. We now describe a t-wise finite coloring of $\mathbb{N}$. Consider $s \in [t]$. Given a set $A \subseteq \mathbb{N}^{(s)}$, $\texttt{col}_s(A)$ is a vector indexed by $(y, \mathbf{x}, \sigma)$, where $y \in D'$, $\mathbf{x} \in D^s$, and σ is a s-permutation, whose entry is $q^y_{\mathbf{x},\sigma}(A)$. The domain is finite, so the number of dimensions is finite. Since the tester is discrete, the number of possible colors entries is finite. Applying Theorem 3, we know the existence of a t-wise monochromatic infinite set $\mathbf{R} \subseteq \mathbb{N}$. We have the property that for any $y, \mathbf{x}, \sigma$, and any two sets $A, B \in \mathbf{R}^{(s)}$, we have $q^y_{\mathbf{x},\sigma}(A) = q^y_{\mathbf{x},\sigma}(B)$. That is, the algorithm $\mathcal{A}$ is a comparison based tester for functions with range $\mathbf{R}$.

Consider the strictly monotone map $\phi : \mathbb{N} \to \mathbf{R}$, where $\phi(b)$ is the bth element of $\mathbf{R}$ in sorted order. Now given any function $f : \mathbf{D} \to \mathbb{N}$, consider the function $\phi \circ f : \mathbf{D} \to \mathbf{R}$. Consider an algorithm $\mathcal{A}'$ which on input f runs $\mathcal{A}$ on $\phi \circ f$. More precisely, whenever $\mathcal{A}$ queries a point x, it gets answer $\phi \circ f(x)$. Observe that if f is monotone (or ε-far from monotone), then so is $\phi \circ f$, and therefore, the algorithm $\mathcal{A}'$ is a $(t, \varepsilon, 2\delta)$-tester of $\phi \circ f$. Since the range of $\phi \circ f$ is $\mathbf{R}$, $\mathcal{A}'$ is comparison-based.

3 Lower Bounds

We assume that n is a power of 2, set $\ell := \log_2 n$, and think of $[n]$ as $\{0, 1, \ldots, n-1\}$. For any number $0 \leq z < n$, we think of the binary representation of z as an ℓ-bit vector $(z_1, z_2, \ldots, z_\ell)$, where z_1 is the least significant bit.

Consider the following canonical, one-to-one mapping $\phi : [n]^d \to \{0,1\}^{d\ell}$. For any $\boldsymbol{y} = (y_1, y_2, \ldots, y_d) \in [n]^d$, we concatenate binary representations of the y_is in order to get a $d\ell$-bit vector $\phi(\boldsymbol{y})$. Hence, we can transform a function $f : \{0,1\}^{d\ell} \to \mathbb{N}$ into a function $\widetilde{f} : [n]^d \to \mathbb{N}$ by defining $\widetilde{f}(\boldsymbol{y}) := f(\phi(\boldsymbol{y}))$.

We will now describe a distribution of functions over the boolean hypercube with equal mass on monotone and ε-far from monotone functions. The key property is that for a function drawn from this distribution, any deterministic comparison based algorithm errs in classifying it with non-trivial probability.

This property will be used in conjunction with the above mapping to get our final lower bound.

3.1 The Hard Distribution

We focus on functions $f : \{0,1\}^m \to \mathbb{N}$. (Eventually, we set $m = d\ell$.) Given any $x \in \{0,1\}^m$, we let $\texttt{val}(x) := \sum_{i=1}^{m} 2^{i-1}x_i$ denote the number for which x is the binary representation. Here, x_1 denotes the least significant bit of x.

For convenience, we let ε be a power of $1/2$. For $k \in \{1, \ldots, \frac{1}{2\varepsilon}\}$, we let

$$S_k := \{x : \texttt{val}(x) \in [2(k-1)\varepsilon 2^m, 2k\varepsilon 2^m - 1) \ \}.$$

Note that S_ks partition the hypercube, with each $|S_k| = \varepsilon 2^{m+1}$. In fact, each S_k is a subhypercube of dimension $m' := m + 1 - \log(1/\varepsilon)$, with the minimal element having all zeros in the m' least significant bits, and the maximal element having all ones in those.

We describe a distribution $\mathcal{F}_{m,\varepsilon}$ on functions. The support of $\mathcal{F}_{m,\varepsilon}$ consists of $f(x) = 2\texttt{val}(x)$ and $\frac{m'}{2\varepsilon}$ functions indexed as $g_{j,k}$ with $j \in [m']$ and $k \in [\frac{1}{2\varepsilon}]$, defined as follows.

$$g_{j,k}(x) = \begin{cases} 2\texttt{val}(x) - 2^j - 1 & \text{if } x_j = 1 \text{ and } x \in S_k \\ 2\texttt{val}(x) & \text{otherwise} \end{cases}$$

The distribution $\mathcal{F}_{m,\varepsilon}$ puts probability mass $1/2$ on the function $f = 2\texttt{val}$ and $\frac{\varepsilon}{m'}$ on each of the $g_{j,k}$s. All these functions take distinct values on their domain. Note that $2\texttt{val}$ induces a total order on $\{0,1\}^m$.

The Distinguishing Problem: Given query access to a random function f from $\mathcal{F}_{m,\varepsilon}$, we want a deterministic comparison-based algorithm that declares that $f = 2\texttt{val}$ or $f \neq 2\texttt{val}$. We refer to any such algorithm as a *distinguisher*. Naturally, we say that the distinguisher errs on f if its declaration is wrong. Our main lemma is the following.

Lemma 2. *Any deterministic comparison-based distinguisher that makes less than* $\frac{m'}{8\varepsilon}$ *queries errs with probability at least* $1/8$.

The following proposition allows us to focus on *non-adaptive* comparison based testers.

Proposition 1. *Given any deterministic comparison-based distinguisher* $\mathcal{A}$ *for* $\mathcal{F}_{m,\varepsilon}$ *that makes at most* t *queries, there exists a deterministic* non-adaptive *comparison-based distinguisher* $\mathcal{A}'$ *making at most* t *queries whose probability of error on* $\mathcal{F}_{m,\varepsilon}$ *is at most that of* $\mathcal{A}$.

Proof. We represent $\mathcal{A}$ as a comparison tree. For any path in $\mathcal{A}$, the total number of distinct domain points involved in comparisons is at most t. Note that $2\texttt{val}(x)$ is a total order, since for any x, y either $\texttt{val}(x) < \texttt{val}(y)$ or vice versa. For any comparison in $\mathcal{A}$, there is an outcome inconsistent with this ordering.

(An outcome "$f(x) < f(y)$" where $\texttt{val}(x) > \texttt{val}(y)$ is inconsistent with the total order.) We construct a comparison tree $\mathcal{A}'$ where we simply reject whenever a comparison is inconsistent with the total order, and otherwise mimics $\mathcal{A}$. The comparison tree of $\mathcal{A}'$ has an error probability at most that of $\mathcal{A}$ (since it may reject a few $f \neq 2\texttt{val}$), and is just a path. Hence, it can be modeled as a non-adaptive distinguisher. We query upfront all the points involving points on this path, and make the relevant comparisons for the output.

Combined with Proposition 1, the following lemma completes the proof of Lemma 2.

Lemma 3. *Any deterministic, non-adaptive, comparison-based distinguisher $\mathcal{A}$ making fewer than $t \leq \frac{m'}{8\varepsilon}$ queries, errs with probability at least $1/8$.*

Proof. Let X be the set of points queried by the distinguisher. Set $X_k =: X \cap S_k$; these form a partition of X. We say that a pair of points (x, y) *captures* the (unique) coordinate j, if j is the largest coordinate where $x_j \neq y_j$. (By largest coordinate, we refer to most significant bit.) For a set Y of points, we say Y captures coordinate j if there is a pair in Y that captures j.

Claim. For any j, k, if the algorithm distinguishes between $\texttt{val}$ and $g_{j,k}$, then X_k captures j.

Proof. If the algorithm distinguishes between $\texttt{val}$ and $g_{j,k}$, there must exist $(x, y) \in X$ such that $\texttt{val}(x) < \texttt{val}(y)$ and $g_{j,k}(x) > g_{j,k}(y)$. We claim that x and y capture j; this will also imply they lie in the same $S_{k'}$ since the $m - j$ most significant bit of x and y are the same.

Firstly, observe that we must have $y_j = 1$ and $x_j = 0$; otherwise, $g_{j,k}(y) - g_{j,k}(x) \geq 2(\texttt{val}(y) - \texttt{val}(x)) > 0$ contradicting the supposition. Now suppose (x, y) don't capture j implying there exists $i > j$ which is the largest coordinate at which they differ. Since $\texttt{val}(y) > \texttt{val}(x)$ we have $y_i = 1$ and $x_j = 0$. Therefore, we have

$$g_{j,k}(y) - g_{j,k}(x) \geq 2(\texttt{val}(y) - \texttt{val}(x)) - 2^j - 1 \geq (2^i + 2^j) - \sum_{1 \leq r < i} 2^r - 2^j - 1 > 0.$$

So, x, y capture j and lie in the same $S_{k'}$. If $k' \neq k$, then again $g_{j,k}(y) - g_{j,k}(x) = 2(\texttt{val}(y) - \texttt{val}(x)) > 0$. Therefore, X_k captures j.

The following claim allows us to complete the proof of the lemma.

Claim. A set Y captures at most $|Y| - 1$ coordinates.

Proof. We prove this by induction on $|Y|$. When $|Y| = 2$, this is trivially true. Otherwise, pick the largest coordinate j captured by Y and let $Y_0 = \{y : y_j = 0\}$ and $Y_1 = \{y : y_j = 1\}$. By induction, Y_0 captures at most $|Y_0| - 1$ coordinates, and Y_1 captures at most $|Y_1| - 1$ coordinates. Pairs $(x, y) \in Y_0 \times Y_1$ only capture coordinate j. Therefore, the total number of captured coordinates is at most $|Y_0| - 1 + |Y_1| - 1 + 1 = |Y| - 1$.

If $|X| \leq m'/8\varepsilon$, then there exist at least $1/4\varepsilon$ values of k such that $|X_k| \leq m'/2$. By the previous claim, each such X_k captures at most $m'/2$ coordinates. Therefore, there exist at least $\frac{1}{4\varepsilon} \cdot \frac{m'}{2} = \frac{m'}{8\varepsilon}$ functions $g_{j,k}$ that are indistinguishable from the monotone function $\texttt{2val}$ to a comparison-based procedure that queries X. This implies the distinguisher must err (make a mistake on either these $g_{j,k}$s or $\texttt{2val}$) with probability at least $\min(\frac{\varepsilon}{m'} \cdot \frac{m'}{8\varepsilon}, 1/2) = 1/8$.

3.2 The Final Bound

Recall, given function $f : \{0,1\}^{d\ell} \to \mathbb{N}$, we have the function $\widetilde{f} : [n]^d \to \mathbb{N}$ by defining $\widetilde{f}(\boldsymbol{y}) := f(\phi(\boldsymbol{y}))$. We start with the following observation.

Proposition 2. *The function* $\widetilde{\texttt{2val}}$ *is monotone and every* $\widetilde{g_{j,k}}$ *is* $\varepsilon/2$*-far from being monotone.*

Proof. Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be elements in $[n]^d$ such that $\boldsymbol{u} \prec \boldsymbol{v}$. We have $\texttt{val}(\phi(\boldsymbol{u})) < \texttt{val}(\phi(\boldsymbol{v}))$, so $\widetilde{\texttt{2val}}$ is monotone. For the latter, it suffices to exhibit a matching of violated pairs of cardinality $\varepsilon 2^{d\ell}$ for $\widetilde{g_{j,k}}$. This is given by pairs $(\boldsymbol{u}, \boldsymbol{v})$ where $\phi(\boldsymbol{u})$ and $\phi(\boldsymbol{v})$ only differ in their jth coordinate, and are both contained in S_k. Note that these pairs are comparable in $[n]^d$ and are violations.

Theorem 4. *Any* $(t, \varepsilon/2, 1/16)$*-monotonicity tester for* $f : [n]^d \to \mathbb{N}$*, must have* $t \geq \frac{d \log n - \log(1/\varepsilon)}{8\varepsilon}$.

Proof. By Theorem 2, it suffices to show this for comparison-based $(t, \varepsilon/2, 1/8)$ testers. By Yao's minimax lemma, it suffices to produce a distribution $\mathcal{D}$ over functions $f : [n]^d \to \mathbb{N}$ such that any deterministic comparison-based $(t, \varepsilon/2, 1/8)$-monotonicity tester for $\mathcal{D}$ must have $t \geq s$, where $s := \frac{d \log n - \log(1/\varepsilon)}{8\varepsilon}$.

Consider the distribution $\mathcal{D}$ where we generate f from $\mathcal{F}_{m,\varepsilon}$ and output $\widetilde{f}$. Suppose $t < s$. By Proposition 2, the deterministic comparison based monotonicity tester acts as a determinisitic comparison-based distinguisher for $\mathcal{F}_{m,\varepsilon}$ making fewer than s queries, contradicting Lemma 3.

4 Conclusion

In this paper, we exhibit a lower bound of $\Omega(\varepsilon^{-1} d \log n - \varepsilon^{-1} \log \varepsilon^{-1})$ queries on adaptive, two-sided monotonicity testers for functions $f : [n]^d \to \mathbb{N}$, matching the upper bound of $O(\varepsilon^{-1} d \log n)$ queries of [15]. Our proof hinged on two things: that for monotonicity on any partial order one can focus on comparison-based testers, and a lower bound on comparison-based testers for the hypercube domain. Some natural questions are left open. Can one focus on some restricted class of testers for the Lipschitz property, and more generally, can one prove adaptive, two-sided lower bounds for the Lipschitz property testing on the hypergrid/cube? Currently, a $\Omega(d \log n)$-query non-adaptive lower bound is known for the problem [17]. Can one prove comparison-based lower bounds

for monotonicity testing on a general N-vertex poset? For the latter problem, there is a $O(\sqrt{N/\varepsilon})$-query non-adaptive tester, and a $\Omega(N^{\frac{1}{\log\log N}})$-query non-adaptive, two-sided error lower bound [7]. Our methods do not yield any results for bounded ranges, but there are significant gaps in our understanding for that regime. For monotonicity testing of boolean functions $f : \{0,1\}^n \to \{0,1\}$, the best adaptive lower bound of $\Omega(\log n)$, while the best non-adaptive bound is $\Omega(\sqrt{n})$ [7].

References

1. Rubinfeld, R., Sudan, M.: Robust characterization of polynomials with applications to program testing. SIAM Journal of Computing 25, 647–668 (1996) 425
2. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. Journal of the ACM 45(4), 653–750 (1998) 425
3. Goldreich, O., Goldwasser, S., Lehman, E., Ron, D., Samorodnitsky, A.: Testing monotonicity. Combinatorica 20, 301–337 (2000) 425, 426
4. Ergun, F., Kannan, S., Kumar, R., Rubinfeld, R., Viswanathan, M.: Spot-checkers. Journal of Computer Systems and Sciences (JCSS) 60(3), 717–751 (2000) 426
5. Dodis, Y., Goldreich, O., Lehman, E., Raskhodnikova, S., Ron, D., Samorodnitsky, A.: Improved testing algorithms for monotonicity. In: Hochbaum, D.S., Jansen, K., Rolim, J.D.P., Sinclair, A. (eds.) RANDOM-APPROX 1999. LNCS, vol. 1671, pp. 97–108. Springer, Heidelberg (1999) 426
6. Lehman, E., Ron, D.: On disjoint chains of subsets. Journal of Combinatorial Theory, Series A 94(2), 399–404 (2001) 426
7. Fischer, E., Lehman, E., Newman, I., Raskhodnikova, S., Rubinfeld, R., Samorodnitsky, A.: Monotonicity testing over general poset domains. In: Proceedings of the 34th Annual ACM Symposium on the Theory of Computing (STOC), pp. 474–483 (2002) 426, 434
8. Ailon, N., Chazelle, B.: Information theory in property testing and monotonicity testing in higher dimension. Information and Computation 204(11), 1704–1717 (2006) 426
9. Fischer, E.: On the strength of comparisons in property testing. Information and Computation 189(1), 107–116 (2004) 426, 427, 428
10. Halevy, S., Kushilevitz, E.: Testing monotonicity over graph products. Random Structures and Algorithms 33(1), 44–67 (2008) 426
11. Parnas, M., Ron, D., Rubinfeld, R.: Tolerant property testing and distance approximation. Journal of Computer and System Sciences 72(6), 1012–1042 (2006) 426
12. Ailon, N., Chazelle, B., Comandur, S., Liu, D.: Estimating the distance to a monotone function. Random Structures and Algorithms 31(3), 1704–1711 (2006) 426
13. Batu, T., Rubinfeld, R., White, P.: Fast approximate PCPs for multidimensional bin-packing problems. Information and Computation 196(1), 42–56 (2005) 426
14. Blais, E., Brody, J., Matulef, K.: Property testing lower bounds via communication complexity. Computational Complexity 21(2), 311–358 (2012) 426
15. Chakrabarty, D., Seshadhri, C.: Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In: Proceedings of Symposium on Theory of Computing, STOC (2013) 426, 433

16. Brody, J.: Personal communication (2013) 426
17. Blais, E., Raskhodnikova, S., Yaroslavtsev, G.: Lower bounds for testing properties of functions on hypergrid domains. Technical Report TR13-036, ECCC (March 2013) 426, 427, 433
18. Briët, J., Chakraborty, S., García-Soriano, D., Matsliah, A.: Monotonicity testing and shortest-path routing on the cube. Combinatorica 32(1), 35–53 (2012) 426
19. Bollobás, B.: Modern Graph Theory. Springer (2000) 429

Small-Bias Sets for Nonabelian Groups
Derandomizations of the Alon-Roichman Theorem

Sixia Chen[1], Cristopher Moore[2], and Alexander Russell[1]

[1] University of Connecticut
acr@cse.uconn.edu, chensixia09@gmail.com
[2] The Santa Fe Institute
moore@santafe.edu

Abstract. In analogy with ε-biased sets over $\mathbb{Z}_2^n$, we construct explicit ε-biased sets over nonabelian finite groups G. That is, we find sets $S \subset G$ such that $\| \mathbb{E}_{x \in S} \rho(x) \| \leq \varepsilon$ for any nontrivial irreducible representation ρ. Equivalently, such sets make G's Cayley graph an expander with eigenvalue $|\lambda| \leq \varepsilon$. The Alon-Roichman theorem shows that random sets of size $O(\log |G|/\varepsilon^2)$ suffice. For groups of the form $G = G_1 \times \cdots \times G_n$, our construction has size $\mathrm{poly}(\max_i |G_i|, n, \varepsilon^{-1})$, and we show that a specific set $S \subset G^n$ considered by Meka and Zuckerman that fools read-once branching programs over G is also ε-biased in this sense. For solvable groups whose abelian quotients have constant exponent, we obtain ε-biased sets of size $(\log |G|)^{1+o(1)} \, \mathrm{poly}(\varepsilon^{-1})$. Our techniques include derandomized squaring (in both the matrix product and tensor product senses) and a Chernoff-like bound on the expected norm of the product of independently random operators that may be of independent interest.

1 Introduction

Small-bias sets are useful combinatorial objects for derandomization, and are particularly well-studied over the Boolean hypercube $\{0,1\}^n$. Specifically, if we identify the hypercube with the group $\mathbb{Z}_2^n$, then a *character* χ is a homomorphism from $\mathbb{Z}_2^n$ to $\mathbb{C}$. We say that a set $S \subseteq \mathbb{F}_2^n$ is *ε-biased* if, for all characters χ, $|\mathbb{E}_{x \in S} \chi(x)| \leq \varepsilon$, except for the trivial character $\mathbb{1}$, which is identically equal to 1. Since any character of $\mathbb{F}_2^n$ can be written $\chi(x) = (-1)^{k \cdot x}$ where $k \in \mathbb{Z}_2^n$ is the "frequency vector," this is equivalent to the familiar definition which demands that on any nonempty set of bits, x's parity should be odd or even with roughly equal probability, $(1 \pm \varepsilon)/2$.

It is easy to see that ε-biased sets of size $O(n/\varepsilon^2)$ exist: random sets suffice. Moreover, several efficient deterministic constructions are known [13, 2, 3, 4] of size polynomial in n and $1/\varepsilon$. These constructions have been used to derandomize a wide variety of randomized algorithms, replacing random sampling over all of $\{0,1\}^n$ with deterministic sampling on S (see, e.g., [5]). In particular, sampling a function on an ε-biased set yields a good estimate of its expectation if its Fourier spectrum has bounded ℓ_1 norm.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 436–451, 2013.

The question of whether similar constructions exist for nonabelian groups has been a topic of intense interest. Given a group G, a *representation* is a homomorphism ρ from G into the group $\mathsf{U}(d)$ of $d \times d$ unitary matrices for some $d = d_\rho$. If G is finite, then up to isomorphism there is a finite set $\widehat{G}$ of *irreducible* representations, or *irreps* for short, such that any representation σ can be written as a direct sum of irreps. These irreps form the basis for harmonic analysis over G, analogous to classic discrete Fourier analysis on abelian groups such as $\mathbb{Z}_p$ or $\mathbb{Z}_2^n$.

Generalizing the standard notion from characters to matrix-valued representations, we say that a set $S \subseteq G$ is ε-*biased* if, for all nontrivial irreps $\rho \in \widehat{G}$, $\left\|\mathbb{E}_{x \in S}\, \rho(x)\right\| \leq \varepsilon$, where $\|\cdot\|$ denotes the operator norm. There is a natural connection with expander graphs. If we define a Cayley graph on G using S as a set of generators, then G becomes an expander if and only if S is ε-biased. Specifically, if M is the stochastic matrix equal to $1/|S|$ times the adjacency matrix, corresponding to the random walk where we multiply by a random element of S at each step, then M's second eigenvalue has absolute value ε. Thus ε-biased sets S are precisely sets of generators that turn G into an expander of degree $|S|$.

The Alon-Roichman theorem [1] asserts that a uniformly random set of $O((\log |G|)/\varepsilon^2)$ group elements is ε-biased with high probability. Thus, our goal is to derandomize the Alon-Roichman theorem—finding explicit constructions of ε-biased sets of size polynomial in $\log |G|$ and $1/\varepsilon$. (For another notion of derandomizing the Alon-Roichman theorem, in time $\mathrm{poly}(|G|)$, see Wigderson and Xiao [16].)

Throughout, we apply the technique of "derandomized squaring"—analogous to the principal construction in Rozenman and Vadhan's alternate proof of Reingold's theorem [15] that Undirected Reachability is in $\mathsf{LOGSPACE}$. In particular, we observe that derandomized squaring provides a generic amplification tool in our setting; specifically, given a constant-bias set S, we can obtain an ε-biased set of size $O(|S|\varepsilon^{-11})$. We also use a tensor product version of derandomized squaring to build ε-biased sets for G recursively, from ε-biased sets for its subgroups or quotients.

Homogeneous direct products and branching programs. Groups of the form G^n where G is fixed have been actively studied by the pseudorandomness community as a specialization of the class of constant-width branching programs. The problem of fooling "read-once" group programs induces an alternate notion of ε-biased sets over groups of the form G^n defined by Meka and Zuckerman [10]. Specifically, a read-once branching program on G consists of a tuple $\mathbf{g} = (g_1, \ldots, g_n) \in G^n$ and takes a vector of n Boolean variables $\mathbf{b} = (b_1, \ldots, b_n)$ as input. At each step, it applies $g_i^{b_i}$, i.e., g_i if $b_i = 1$ and 1 if $b_i = 0$. They say a set $S \subset G^n$ is ε-biased if, for all $\mathbf{b} \neq 0$, the distribution of $\mathbf{g}^{\mathbf{b}}$ is close to uniform, i.e.,

$$\forall h \in G : \left| \Pr_{\mathbf{g} \in S}\left[\mathbf{g}^{\mathbf{b}} = h\right] - \frac{1}{|G|} \right| \leq \varepsilon \quad \text{where} \quad \mathbf{g}^{\mathbf{b}} = \prod_{i=1}^{n} g_i^{b_i}\,. \tag{1}$$

As they comment, there is no obvious relationship between this definition and the one we consider. However, we show in Section 2 that a particular set shown to have property (1) in [10] is also ε-biased in our sense; the proof is completely different. This yields ε-biased sets of size $O(n \cdot \mathrm{poly}(\varepsilon^{-1}))$.

Inhomogeneous direct products. For the more general case of groups of the form $G = G_1 \times \cdots \times G_n$, we show that a tensor product adaptation of derandomized squaring yields a recursive construction of ε-biased sets of size polynomial in $\max_i |G_i|$, n, and $1/\varepsilon$.

Normal extensions and "smoothly solvable" groups. Finally, we show that if G is solvable and has abelian quotients of bounded exponent, we can construct ε-biased sets of size $(\log |G|)^{1+o(1)} \mathrm{poly}(\varepsilon^{-1})$. Here we use the representation theory of solvable groups to build an ε-biased set for G recursively from those for a normal subgroup H and the quotient G/H.

2 An Explicit Set for G^n with Constant ε

Meka and Zuckerman [10] considered the following construction for fooling read-once group branching programs:

Definition 1. *Let G be a group and $n \in \mathbb{N}$. Then, given an ε-biased set S over $\mathbb{Z}_{|G|}^n$, define $T_S \triangleq \{(g^{s_1}, \ldots, g^{s_n}) \mid g \in G, (s_1, \ldots, s_n) \in S\}$.*

We prove the following theorem, showing that this construction yields sets of small bias in our sense (and, hence, expander Cayley graphs over G^n).

Theorem 1. *If S is ε-biased over $\mathbb{Z}_{|G|}^n$ then T_S is $(1 - \Omega(1/\log\log|G|)^2 + \varepsilon)$-biased over G^n.*

Anticipating the proof, we set down the following definition.

Definition 2. *Let G be a finite group and $\widehat{G}$ be the set of equivalence classes of irreducible unitary representations of G. For a representation $\rho \in \widehat{G}$ and a subgroup H, define*

$$\Pi_H^\rho \triangleq \mathop{\mathbb{E}}_{h \in H} \rho(h)$$

to be the projection operator induced by the subgroup H in ρ. In the case where $H = \langle g \rangle$ is the cyclic group generated by g, we use the following shorthand:

$$\Pi_g^\rho = \Pi_{\langle g \rangle}^\rho .$$

Finally, for groups of the form G^n we use the following convention. Recall that any irreducible representation $\bar{\rho} \in \widehat{G^n}$ is a tensor product, $\bar{\rho} = \bigotimes_{i=1}^n \rho_i$ where $\rho_i \in \widehat{G}$ for each i. That is, if $\bar{g} = (g_1, \ldots, g_n)$, then $\bar{\rho}(\bar{g}) = \bigotimes_{i=1}^n \rho_i(g_i)$. Then for an element $g \in G$, we write

$$\Pi_g^{\bar{\rho}} \triangleq \Pi_{\langle g \rangle^n}^{\bar{\rho}} = \bigotimes_{i=1}^n \Pi_g^{\rho_i} \tag{2}$$

for the projection operator determined by the abelian subgroup $\langle g \rangle^n$.

Lemma 1. *Let G be a finite group and ρ a nontrivial irreducible representation of G. Then $\left\|\mathbb{E}_{g\in G} \Pi_g^\rho\right\| \le 1 - \phi(|G|)/|G| \le 1 - \Omega\left(1/\log\log|G|\right)$, where $\phi(\cdot)$ denotes the Euler totient function.*

Proof. Expanding the definition of $\Pi_{\langle g\rangle}^\rho$, we have

$$\left\|\mathop{\mathbb{E}}_{g\in G} \Pi_g^\rho\right\| = \left\|\mathop{\mathbb{E}}_{g\in G}\mathop{\mathbb{E}}_{t\in\mathbb{Z}_{|G|}} \rho(g^t)\right\| \le \mathop{\mathbb{E}}_{t\in\mathbb{Z}_{|G|}}\left\|\mathop{\mathbb{E}}_{g} \rho(g^t)\right\|.$$

Recall that the function $x \mapsto x^k$ is a bijection in any group G for which $\gcd(|G|, k) = 1$. Moreover, for such k, $\mathbb{E}_g\, \rho(g^k) = \mathbb{E}_g\, \rho(g) = 0$ as $\rho \neq 1$. Assuming pessimistically that $\|\mathbb{E}_g\, \rho(g^k)\| = 1$ for all other k yields the bound $\|\mathbb{E}_{g\in G}\, \Pi_{\langle g\rangle}^\rho\| \le 1 - \phi(|G|)/|G|$ promised in the statement of the lemma. The function $\phi(n)$ has the property that $\phi(n) > n/(e^\gamma \log\log n + 3/\log\log n)$ for $n > 3$, where $\gamma \approx .5772\ldots$ is the Euler constant [14]; this yields the second estimate in the statement of the lemma.

Our proof will rely on the following tail bound for products of operator-valued random variables, proved in Appendix B.

Theorem 2. *Let $\boldsymbol{P}(H)$ denote the cone of positive operators on the Hilbert space H and let $P_1, \ldots, P_k$ be independent random variables taking values in $\boldsymbol{P}(H)$ for which $\|P_i\| \le 1$ and $\left\|\mathbb{E}[P_i]\right\| \le 1 - \delta$. Then for any $\Delta \ge 0$,*

$$\Pr\left[\left\|P_k \cdots P_1\right\| \ge \sqrt{\dim H}\exp\left(-\frac{k\delta}{2} + \Delta\right)\right] \le \dim H \cdot \exp\left(-\frac{\Delta^2}{2k\ln 2}\right).$$

In particular, choosing $\Delta = k\delta/3$, we conclude that

$$\Pr\left[\left\|P_k \cdots P_1\right\| \ge \sqrt{\dim H}\exp\left(-\frac{k\delta}{6}\right)\right] \le \dim H \cdot \exp\left(-\frac{k\delta^2}{13}\right).$$

We return to the proof of Theorem 1.

Proof (of Theorem 1). For a non-trivial irrep $\bar\rho = \rho_1 \otimes \cdots \otimes \rho_n \in \widehat{G}^n$, we write

$$\mathop{\mathbb{E}}_{\bar t\in T_S} \bar\rho(\bar t) = \mathop{\mathbb{E}}_{g\in G}\mathop{\mathbb{E}}_{\bar s\in S} \bar\rho(g^{\bar s}) = \mathop{\mathbb{E}}_{g\in G}\mathop{\mathbb{E}}_{\bar s\in S} \left(\mathrm{Res}_{\langle g\rangle^n} \bar\rho\right)(g^{\bar s}),$$

where $\bar s = (s_1, \ldots, s_n)$, $g^{\bar s} = (g^{s_1}, \ldots, g^{s_n})$, and $\mathrm{Res}_H\, \bar\rho$ denotes the restriction of $\bar\rho$ to the subgroup $H \subseteq G^n$. For a particular $g \in G$, we decompose the restricted representation $\mathrm{Res}_{\langle g\rangle^n} \bar\rho$ into a direct sum of irreps of the abelian group $\langle g\rangle^n \cong \mathbb{Z}_{|\langle g\rangle|}^n$. This yields $\mathrm{Res}_{\langle g\rangle^n} \bar\rho = \bigoplus_{\chi\in\widehat{\langle g\rangle^n}} \chi^{\oplus a_\chi}$, where each χ is a one-dimensional representation of the cyclic group $\langle g\rangle^n$ and a_χ denotes the multiplicity with which χ appears in the decomposition.

Now, as S is an ε-biased set over $\mathbb{Z}_{|G|}^n$, its quotient modulo any divisor d of $|G|$ is ε-biased over $\mathbb{Z}_d^n$. It follows that $|\mathbb{E}_{\bar s\in S}\, \chi(\bar s)| \le \varepsilon$ for any nontrivial χ; when χ is trivial, the expectation is 1. Thus for any fixed $g \in G$ we may write $\mathbb{E}_{\bar s\in S} \left(\mathrm{Res}_{\langle g\rangle^n} \bar\rho\right)(g^{\bar s}) = \Pi_g^{\bar\rho} + E_g^{\bar\rho}$. Recall that $\Pi_g^{\bar\rho}$ is the projection operator

onto the space associated with the copies of the trivial representation of $\langle g \rangle^n$ in $\mathrm{Res}_{\langle g \rangle^n} \bar{\rho}$, i.e., the expectation we would obtain if $\bar{s}$ ranged over all of $\langle g \rangle^n$ instead over just S. The "error operator" $E_g^{\bar{\rho}}$ arises from the nontrivial representations of $\langle g \rangle^n$ appearing in $\mathrm{Res}_{\langle g \rangle^n} \bar{\rho}$, and has operator norm bounded by ε. It follows that

$$\begin{aligned} \left\| \mathop{\mathbb{E}}_{\bar{t} \in T} \bar{\rho}(\bar{t}) \right\| &= \left\| \mathop{\mathbb{E}}_{g \in G} \left(\mathop{\mathbb{E}}_{\bar{s} \in S} \bar{\rho}(g^{\bar{s}}) \right) \right\| = \left\| \mathop{\mathbb{E}}_{g \in G} \left(\Pi_g^{\bar{\rho}} + E_g^{\bar{\rho}} \right) \right\| \\ &\leq \left\| \mathop{\mathbb{E}}_{g \in G} \Pi_g^{\bar{\rho}} \right\| + \left\| \mathop{\mathbb{E}}_{g \in G} E_g^{\bar{\rho}} \right\| \leq \left\| \mathop{\mathbb{E}}_{g \in G} \Pi_g^{\bar{\rho}} \right\| + \varepsilon \,, \end{aligned}$$

and it remains to bound $\| \mathbb{E}_{g \in G} \Pi_g^{\bar{\rho}} \|$.

As orthogonal projections are Hermitian, $\mathbb{E}_{g \in G} \Pi_g^{\bar{\rho}}$ is Hermitian, and for any positive k we have

$$\left\| \mathop{\mathbb{E}}_{g \in G} \Pi_g^{\bar{\rho}} \right\| = \sqrt[k]{\left\| \left(\mathop{\mathbb{E}}_{g \in G} \Pi_g^{\bar{\rho}} \right)^k \right\|}, \tag{3}$$

so we focus on the operator $\left(\mathbb{E}_{g \in G} \Pi_g^{\bar{\rho}} \right)^k$. Expanding $\Pi_g^{\bar{\rho}} = \bigotimes_i \Pi_g^{\rho_i}$, we may write

$$\left(\mathop{\mathbb{E}}_{g \in G} \Pi_g^{\bar{\rho}} \right)^k = \mathop{\mathbb{E}}_{g_1, \ldots, g_k} \left[\Pi_{g_1}^{\bar{\rho}} \cdots \Pi_{g_k}^{\bar{\rho}} \right] = \mathop{\mathbb{E}}_{g_1, \ldots, g_k} \left[\bigotimes_{i=1}^{n} \Pi_{g_1}^{\rho_i} \cdots \Pi_{g_k}^{\rho_i} \right] . \tag{4}$$

As $\bar{\rho}$ is nontrivial, there is some coordinate j for which ρ_j is nontrivial. Combining (4) with the fact that $\|A \otimes B\| = \|A\| \|B\|$, we conclude that

$$\left\| \left(\mathop{\mathbb{E}}_{g \in G} \Pi_g^{\bar{\rho}} \right)^k \right\| \leq \mathop{\mathbb{E}}_{g_1, \ldots, g_k} \left\| \bigotimes_{i=1}^{n} \Pi_{g_1}^{\rho_i} \cdots \Pi_{g_k}^{\rho_i} \right\| \leq \mathop{\mathbb{E}}_{g_1, \ldots, g_k} \left\| \Pi_{g_1}^{\rho_j} \cdots \Pi_{g_k}^{\rho_j} \right\| . \tag{5}$$

Lemma 1 asserts that $\| \mathbb{E}_g \Pi_g^{\rho_j} \| \leq 1 - \delta_G$, where $\delta_G = \Omega(1/\log\log|G|)$. It follows then from Theorem 2 that

$$\Pr_{g_1, \ldots, g_k} \Big[\underbrace{\left\| \Pi_{g_1}^{\rho_j} \cdots \Pi_{g_k}^{\rho_j} \right\| \geq \sqrt{d_j} \exp(-k\delta_G/6)}_{(\ddagger)} \Big] \leq d_j \cdot \exp\left(-k\delta_G^2/13\right) , \tag{6}$$

where $d_j = \dim \rho_j$. This immediately provides a bound on $\|(\mathbb{E}_g \Pi_g^{\bar{\rho}})^k\|$. Specifically, combining (5) with (6), let us pessimistically assume that $\|\Pi_{g_1}^{\rho_j} \cdots \Pi_{g_k}^{\rho_j}\| = d_j \exp(-k\delta_G/6)$ for tuples $(g_1, \ldots, g_k)$ that do not enjoy property $(\ddagger)$, and 1 for tuples that do. Then

$$\begin{aligned} \left\| \left(\mathop{\mathbb{E}}_{g \in G} \Pi_g^{\bar{\rho}} \right)^k \right\| &\leq \mathop{\mathbb{E}}_{g_1, \ldots, g_k} \left\| \Pi_{g_1}^{\rho_j} \ldots \Pi_{g_k}^{\rho_j} \right\| \\ &\leq d_j \exp\left(-k\delta_G^2/13\right) + \left(1 - d_j \exp\left(-k\delta_G^2/13\right)\right) \sqrt{d_j} \exp\left(-k\delta_G/6\right) \\ &\leq 2 d_j \exp\left(-k\delta_G^2/13\right) , \end{aligned}$$

and hence $\left\| \mathbb{E}_{g \in G} \Pi_g^{\bar{\rho}} \right\| \leq \inf_k \left(\sqrt[k]{2d_j} \right) \cdot \exp(-\delta_G^2/13) = 1 - \Omega\left(1/\log\log|G|\right)^2$, where we take the limit of large k.

3 Derandomized Squaring and Amplification

In this section we discuss how to amplify ε-biased sets in a generic way. Specifically, we use derandomized squaring to prove the following.

Theorem 3. *Let G be a group and S an $1/10$-biased set on G. Then for any $\varepsilon > 0$, there is an ε-biased set S_ε on G of size $O(|S|\varepsilon^{-11})$. Moreover, assuming that multiplication can be efficiently implemented in G, the set S_ε can be constructed from S in time polynomial in $|S_\varepsilon|$.*

We have made no attempt to improve the exponent of ε in $|S_\varepsilon|$.

Our approach is similar to [15]. Roughly, if S is an ε-biased set on G we can place a degree-d expander graph Γ on the elements of S to induce a new set

$$S \times_\Gamma S \triangleq \{st \mid (s,t) \text{ an edge of } \Gamma\}.$$

If $\rho : G \to \mathsf{U}(V)$ is a nontrivial representation of G, by assumption $\| \mathbb{E}_{s \in S} \rho(g)\| \leq \varepsilon$. Applying a natural operator-valued Rayleigh quotient for expander graphs (see Lemma 2 below), we conclude that

$$\left\| \mathop{\mathbb{E}}_{(s,t)\in\Gamma} \rho(s)\rho(t) \right\| = \left\| \mathop{\mathbb{E}}_{(s,t)\in\Gamma} \rho(st) \right\| \leq \lambda(\Gamma) + \varepsilon^2 .$$

If Γ comes from a family of Ramanujan-like expanders, then $\lambda(\Gamma) = \Theta(1/\sqrt{d})$, and we can guarantee that $\lambda(\Gamma) = O(\varepsilon^2)$ by selecting $d = \Theta(\varepsilon^{-4})$. The size of the set then grows by a factor of $|S \times_\Gamma S|/|S| = d = \Theta(\varepsilon^{-4})$. We make this precise in Lemma 3 below, which regrettably loses an additional factor of ε^{-1}.

Preparing for the proof of Theorem 3, we record some related material on expander graphs.

Expanders and derandomized products. For a d-regular graph $G = (V, E)$, let A denote its normalized adjacency matrix: $A_{uv} = 1/d$ if $(u, v) \in E$ and 0 otherwise. Then A is stochastic, normal, and has operator norm $\|A\| = 1$; the uniform eigenvector $\mathbf{y}^+$ given by $y^+_s = 1$ for all $s \in V$ has eigenvalue 1. When G is connected, the eigenspace associated with 1 is spanned by this eigenvector, and all other eigenvalues lie in $[-1, 1)$.

Bipartite graphs will play a special role in our analysis. We write a bipartite graph G on the bipartition U, V as the tuple $G = (U, V; E)$. In a regular bipartite graph, we have $|U| = |V|$ and -1 is an eigenvalue of A associated with the eigenvector $\mathbf{y}^-$ which is $+1$ for $s \in U$ and -1 for $s \in V$. When G is connected, the eigenspace associated with -1 is one-dimensional, and all other eigenvalues lie in $(-1, 1)$: we let $\lambda(G) < 1$ be the leading nontrivial eigenvalue: $\lambda(G) = \sup_{\mathbf{y} \perp \mathbf{y}^\pm} \|M\mathbf{y}\|/\|\mathbf{y}\|$. When $\mathbf{y} \perp \mathbf{y}^\pm$, observe that $|\langle \mathbf{y}, M\mathbf{y}\rangle| \leq \|\mathbf{y}\| \cdot \|M\mathbf{y}\| \leq \lambda\|\mathbf{y}\|^2$ by Cauchy-Schwarz.

We say that a d-regular, connected, bipartite graph $G = (U, V; E)$ for which $|U| = |V| = n$ and $\lambda(G) \leq \Lambda$ is a *bipartite* (n, d, Λ)-*expander*. A well-known consequence of expansion is that the "Rayleigh quotient" determined by the expander is bounded: for any function $f : U \cup V \to \mathbb{R}$ defined on the vertices of a

(n, d, λ) expander for which $\sum_{u \in U} f(u) = \sum_{v \in V} f(v) = 0$, $\mathbb{E}_{(u,v) \in E} f(u) f(v) \leq \lambda \|f\|_2^2$. We will apply a version of this property pertaining to operator-valued functions.

Lemma 2. *Let $G = (U, V; E)$ be a bipartite (n, d, λ)-expander. Associate with each vertex $s \in U \cup V$ a linear operator X_s on the vector space $\mathbb{C}^d$ such that $\|X_s\| \leq 1$, $\left\|\mathbb{E}_{u \in U} X_u\right\| \leq \varepsilon_U$, and $\left\|\mathbb{E}_{v \in V} X_v\right\| \leq \varepsilon_V$. Then*

$$\left\| \mathop{\mathbb{E}}_{(u,v) \in E} X_u X_v \right\| \leq \lambda + (1 - \lambda) \varepsilon_U \varepsilon_V \, .$$

We will sometimes apply Lemma 2 to the tensor product of operators. That is, given the same assumptions, we have $\left\|\mathbb{E}_{(u,v) \in E} X_u \otimes X_v\right\| \leq \lambda + (1 - \lambda) \varepsilon_U \varepsilon_V$. To see this, simply apply the lemma to the operators $X_u \otimes \mathbb{1}$ and $\mathbb{1} \otimes X_v$.

Critical in our setting is the fact that this conclusion is independent of the dimension d. A proof of this folklore lemma appears in Appendix A; see also [6] for a related application to branching programs over groups.

Amplification. We return now to the problem of amplifying ε-biased sets over general groups.

Lemma 3. *Let S be an ε-biased set on the group G. Then there is an ε'-biased set S' on G for which $\varepsilon' \leq 5\varepsilon^2$ and $|S'| \leq C|S|\varepsilon^{-5}$, where C is a universal constant. Moreover, assuming that multiplication can be efficiently implemented in G, the set S' can be constructed from S in time polynomial in $|S'|$.*

Proof. We proceed as suggested above. The only wrinkle is that we need to introduce an expander graph on the elements of S that achieves second eigenvalue $\Theta(\varepsilon^2)$.

We apply the explicit family of Ramanujan graphs due to Lubotzky, Phillips, and Sarnak [9]. For each pair of primes p and q congruent to 1 modulo 4, they obtain a graph $\Gamma_{p,q}$ with $p(p^2-1)$ vertices, degree $q+1$, and $\lambda(\Gamma_{p,q}) = 2\sqrt{q}/(q+1) < 2/\sqrt{q}$. We treat $\Gamma_{p,q}$ as a bipartite graph by taking the double cover: this introduces a pair of vertices, v_{A} and v_{B}, for each vertex v of $\Gamma_{p,q}$ and introduces an edge (u_A, v_B) for each edge (u, v). This graph has eigenvalues $\pm\lambda$ for each eigenvalue λ of $\Gamma_{p,q}$, so except for the ± 1 eigenspace the spectral radius is unchanged.

As we do not have precise control over the number of vertices in this expander family, we will use a larger graph and approximately tile each side with copies of S. Specifically, we select the smallest primes $p, q \equiv 1 \pmod 4$ for which

$$p(p^2 - 1) > |S| \cdot \lceil \varepsilon^{-1} \rceil \quad \text{and} \quad 2/\sqrt{q} \leq \varepsilon^2 \, . \tag{7}$$

We now associate elements of S with the vertices (of each side) of $\Gamma = \Gamma_{p,q} = (U, V; E)$ as uniformly as possible; specifically, we partition the vertices of U and V into a family of blocks, each of size $|S|$; this leaves a set of less than $|S|$ elements uncovered on each side. Then elements in the blocks are directly associated with elements of S; the "uncovered" elements may in fact be assigned arbitrarily. As $|U| = |V| \geq |S| \lceil \varepsilon^{-1} \rceil$, the uncovered elements above comprise less than an ε-fraction of the vertices. As above, we define the set $S \times_\Gamma S \triangleq \{uv \mid (u, v) \in E\}$

(where we blur the distinction between a vertex and the element of S to which it has been associated).

Consider, finally, a nontrivial representation ρ of G. As the average over any block of U or V has operator norm no more than ε, and we have an ε-fraction of uncovered elements, the average of ρ over each of U and V is no more than $(1-\varepsilon)\varepsilon + \varepsilon \le 2\varepsilon$. Applying Lemma 2, we conclude that $\|\mathbb{E}_{s\in S\times_\Gamma S}\rho(s)\| \le (2\varepsilon)^2 + \lambda(\Gamma) \le 5\varepsilon^2$ by our choice of q (the degree less one).

By Dirichlet's theorem on the density of primes in arithmetic progressions, p and q need be no more than (say) a constant factor larger than the lower bounds $p(p^2-1) > |S|\varepsilon^{-1}$ and $q \ge 4\varepsilon^{-4}$ implied by (7). Thus there is a constant C such that $|S'| = p(p^2-1)(q+1) \le C|S|\cdot\varepsilon^{-5}$.

Remarks. The construction above is saddled with the tasks of identifying appropriate primes p and q, and constructing the generators for the associated expander of [9]. While these can clearly be carried out in time polynomial in $|S'|$, alternate explicit constructions of expander graphs [12] can significantly reduce this overhead. However, no known explicit family of Ramanujan graphs appears to provide enough density to avoid the tiling construction above. On the other hand, expander graphs with significantly weaker properties would suffice for the construction: any uniform bound of the form $\lambda \le c\sqrt{\text{degree}}$ would be enough.

Proof (of Theorem 3). We apply Lemma 3 iteratively. Set $\varepsilon_0 = 1/10$. After t applications, we have an ε_t-biased set where $\varepsilon_t = 2^{-2^t}/5$. After $t = \lceil \log_2 \log_2(1/5\varepsilon)\rceil$ steps, we have $5\varepsilon^2 \le \varepsilon_t \le \varepsilon$. The total increase in size is

$$\frac{|S_\varepsilon|}{|S|} = C^t \left(\prod_{i=0}^{t-1}\varepsilon_i\right)^{-5} = C^t\left(\frac{2\varepsilon_t}{5^{t-1}}\right)^{-5} \le (C/5)^t(50\varepsilon^2)^{-5}$$
$$= O\big(\varepsilon^{-10}(\log\varepsilon^{-1})^{O(1)}\big) = O(\varepsilon^{-11})\,.$$

Combining Theorem 3 with the ε-biased sets constructed in Section 2 we establish a family of ε-biased set over G^n for smaller ε:

Theorem 4. *Fix a group G. There is an ε-biased set in G^n of size $O(n\varepsilon^{-11})$ that can be constructed in time polynomial in n and ε^{-1}.*

Proof. Alon et al. [2] construct a families of explicit codes over finite fields which, in particular, offer δ-biased sets over $\mathbb{Z}_p^n$ of size $O(n)$ for any constant δ. As G is fixed, applying Theorem 1 to these sets over $\mathbb{Z}_{|G|}$ with sufficiently small $\delta \approx (1/\log\log|G|)^2$ yields an ε_0-biased set S_0 over G^n, where ε_0 is a constant close to one (depending on the size of G and the constant δ). We cannot directly apply Theorem 3 to S_0, as the bias may exceed $1/10$. To bridge this constant gap (from ε_0 to $1/10$), we apply the construction of the proof of Theorem 3 with a slight adaptation. Selecting a small constant α, we may enlarge the expander graph to ensure that it has size at least $|S_0|(1/\alpha)$; then the resulting error guarantee on each side of the graph bipartition is no more than $\alpha + (1-\alpha)\varepsilon$ and the product

set has bias no more than $(\alpha + \varepsilon)^2 + \lambda(\Gamma)$. This can be brought as close as desired to ε^2 with appropriate selection of the constants α and $\lambda(G)$. As $\lambda(G)$ is constant, this transformation likewise increases the size of the set by a constant, and this method can reduce the error to $1/10$, say, with a constant-factor penalty in the size of S_0. At this point, Theorem 3 applies, and establishes the bound of the theorem.

4 Inhomogeneous Direct Products

Groups of the form $G = G_1 \times \cdots \times G_n$ appear to frustrate natural attempts to borrow ε-biased sets directly from abelian groups as we did for G^n in Section 2. In this section, we build an ε-biased set for groups of this form by iterating a construction that takes ε-biased sets on two groups G_1 and G_2 and stitches them together, again with an expander graph, to produce an ε'-biased set on $G_1 \times G_2$. In essence, we again use derandomized squaring, but now for the tensor product of two operators rather than their matrix product.

Construction 1. *Let G_1 and G_2 be two groups; for each $i = 1, 2$, let S_i be an ε_i-biased set on G_i. We assume that $|S_1| \leq |S_2|$. Let $\Gamma = (U, V; E)$ be a bipartite $(|S_2|, d, \lambda)$-expander. Associate elements of V with elements of S_2 and, as in the proof of Lemma 3, associate elements of S_1 with U as uniformly as possible. As above, we order the elements of U and tile them with copies of S_1, leaving a collection of no more than S_1 vertices "uncovered"; these vertices are then assigned to an initial subset of S_1 of appropriate size. Define $S_1 \otimes_\Gamma S_2 \subset G_1 \times G_2$ to be the set of edges of Γ (realized as group elements according to the association above).*

Recall that an irreducible representation ρ of $G_1 \times G_2$ is a tensor product $\rho_1 \otimes \rho_2$, where each ρ_i is an irrep of G_i and $\rho(g_1, g_2) = \rho_1(g_1) \otimes \rho_2(g_2)$. If ρ is nontrivial, then one or both of ρ_1 and ρ_2 is nontrivial, and the bias we achieve on ρ will depend on which of these is the case.

Claim. Assuming that $|S_1| \leq |S_2|$, the set $S_1 \otimes_\Gamma S_2$ of Construction 1 has size $d|S_2|$ and bias no more than

$$\max\left(\varepsilon_2, \varepsilon_1 + \frac{|S_1|}{|S_2|}, \lambda + \varepsilon_2\left(\varepsilon_1 + \frac{|S_1|}{|S_2|}\right)\right).$$

Proof. The size bound is immediate. As for the bias, let $\rho = \rho_1 \otimes \rho_2$ be nontrivial. If $\rho_1 = \mathbb{1}$,

$$\left\| \mathop{\mathbb{E}}_{s \in S_1 \otimes_\Gamma S_2} (\rho_1 \otimes \rho_2)(s) \right\| = \left\| \mathop{\mathbb{E}}_{v \in V} \rho_2(v) \right\| \leq \varepsilon_2 \,, \tag{8}$$

as S_2 is in one-to-one correspondence with V. In contrast, if $\rho_2 = \mathbb{1}$, the best we can say is that

$$\left\| \mathop{\mathbb{E}}_{s \in S_1 \otimes_\Gamma S_2} (\rho_1 \otimes \rho_2)(s) \right\| = \left\| \mathop{\mathbb{E}}_{u \in U} \rho_1(u) \right\| \leq \left(1 - \frac{|S_1|}{|S_2|}\right)\varepsilon_1 + \frac{|S_1|}{|S_2|} \leq \varepsilon_1 + \frac{|S_1|}{|S_2|} \tag{9}$$

as in the proof of Lemma 3. When both ρ_i are nontrivial, applying Lemma 2 to (8) and (9) implies that

$$\left\| \mathop{\mathbb{E}}_{s \in S_1 \otimes_\Gamma S_2} (\rho_1 \otimes \rho_2)(s) \right\| \leq \lambda + \varepsilon_2 \left(\varepsilon_1 + \frac{|S_1|}{|S_2|} \right), \tag{10}$$

as desired.

Finally, we apply Construction 1 to groups of the form $G_1 \times \cdots \times G_n$.

Theorem 5. *Let $G = G_1 \times \cdots \times G_n$. Then, for any ε, there is an ε-biased set in G of size* $\mathrm{poly}(\max_i |G_i|, n, \varepsilon^{-1})$. *Furthermore, the set can be constructed in time polynomial in its size.*

Proof. Given the amplification results of Section 3, we may focus on constructing sets of constant bias. We start by adopting the entire group G_i as a 0-biased set for each G_i, and then recursively apply Construction 1. This process will only involve expander graphs of constant degree, which simplifies the task of finding the expander required for Construction 1. In this case, one can construct a constant degree expander graph of desired constant spectral gap on a set X by covering the vertices of X with a family of overlapping expander graphs, uniformizing the degree arbitrarily, and forming a small power of the result. So long as the pairwise intersections of the covering expanders are not too small, the resulting spectral gap can be controlled uniformly. (This luxury was not available to us in the proof of Lemma 3, since in that setting we required λ tending to zero, and insisted on a Ramanujan-like relationship between λ and the degree.)

The recursive construction proceeds by dividing G into two factors: $A = G_1 \times \cdots \times G_{n'}$ and $B = G_{n'+1} \times \cdots \times G_n$, where $n' = \lceil n/2 \rceil$. Given small-biased sets S_A and S_B, we combine them using Construction 1. Examining Claim 4, we wish to ensure that $|S_A|/|S_B|$ is a small enough constant. To arrange for this, we assume without loss of generality that $|S_B| \geq |S_A|$ and duplicate S_B five times, resulting in a (multi-)set S'_B such that $|S_A|/|S'_B| \leq 1/5$.

Assume that each of the recursively constructed sets S_A, S_B has bias at most $1/4$. We apply Construction 1 to S_A and S'_B with an expander Γ of degree d for which $\lambda \leq 1/8$, producing the set $S = S_A \otimes_\Gamma S'_B$. Ideally, we would like S to also be $1/4$-biased, in which case a set of constant bias and size $\mathrm{poly}(\max_i |G_i|, n)$ would follow by induction.

Let $\rho = \rho_A \otimes \rho_B$ be nontrivial, where $\rho_A \in \widehat{A}$ and $\rho_B \in \widehat{B}$. If $\rho_A = \mathbb{1}$ then, as in (8), $\left\| \mathbb{E}_{s \in S} \rho(s) \right\| \leq 1/4$. Likewise, if both ρ_A and ρ_B are nontrivial, (10) gives $\left\| \mathbb{E}_{s \in S} \rho(s) \right\| \leq 1/8 + 1/4(1/4 + 1/5) \leq 1/4$. At first inspection, the case where $\rho_B = \mathbb{1}$ appears problematic, as (9) only provides the discouraging estimate $\left\| \mathbb{E}_{s \in S} \rho(s) \right\| \leq 1/4 + 1/5$. Thus it seems possible that iterative application of Construction 1 could lose control of the error. However, as long as the tiling of U, the left side of the expander in Construction 1, is carried out in a way that ensures that the uncovered elements of U are tiled with respect to *previous* stages of the recursive construction, it is easy to check that subsequent recursive appearances

of this case can contribute no more than the geometric series $1/5+(1/5)^2+\cdots = 1/4$ to the bias. Any following recursive application of the construction in which the representation is nontrivial in both blocks will then drive the error back to $1/4$, as $1/8 + 1/4(1/4 + 1/4) = 1/4$. (If this case occurs at the last stage of recursion, then S still has bias at most $1/4 + 1/5 \leq 1/2$.)

Recall that for the base case of the induction, we treat each G_i as a 0-biased set for itself. Since there are $\log_2 n$ layers of recursion, and each layer multiplies the size of the set by the constant factor $5d$, we end with a $1/2$-biased set S of size at most $(5d)^{\log_2 n} \max_i |G_i| = \mathrm{poly}(\max_i |G_i|, n)$. Finally, applying the amplification of Theorem 3, after first driving the bias down to $1/10$ as in Theorem 4, completes the proof.

We note that if the G_i are of polynomial size, then we can use the results of Wigderson and Xiao [16] to find ε-biased sets of size $O(\log |G_i|)$ in time $\mathrm{poly}(|G_i|)$. Using these sets in the base case of our recursion then gives a ε-biased set for G of size $\mathrm{poly}(\max_i \log |G_i|, n, \varepsilon^{-1})$.

5 Normal Extensions and Smoothly Solvable Groups

While applying these techniques to arbitrary groups (even in the case when they have plentiful subgroups) seems difficult, for solvable groups one can again use a form of derandomized squaring. First, recall the derived series: if G is solvable, then setting $G^{(0)} = G$ and taking commutator subgroups $G^{(i+1)} = [G^{(i)}, G^{(i)}]$ gives a series of normal subgroups, $1 = G^{(\ell)} \lhd \cdots \lhd G^{(1)} \lhd G^{(0)} = G$. We say that ℓ is the *derived length* of G. Each factor $G^{(i)}/G^{(i+1)} = A_i$ is abelian, and $G^{(i)}$ is normal in G for all i. Since $|A_i| \geq 2$, it is obvious that $\ell = O(\log |G|)$. However, more is true. The *composition series* is a refinement of the derived series where each quotient is a cyclic group of prime order, and the length c of this refined series is the *composition length*. Clearly $c \leq \log_2 |G|$. Glasby [8] showed that $\ell \leq 3 \log_2 c + 9 = O(\log c)$, so $\ell = O(\log \log |G|)$.

We focus on groups that are *smoothly solvable* [7], in the sense that the abelian factors have constant exponent. (Their definition of smooth solvability allows the factors to be somewhat more general, but we avoid that here for simplicity.) We then have the following:

Theorem 6. *Let G be a solvable group, and let its abelian factors be of the form $A_i = \mathbb{Z}_{p_i}^t$ (or factors of such groups) where $p_i = O(1)$. Then G possesses an ε-biased set S_ε of size $(\log |G|)^{1+o(1)} \mathrm{poly}(\varepsilon^{-1})$.*

We deliberately gloss over the issue of explicitness. However, we claim that if G is polynomially uniform in the sense of [11], so that we can efficiently express group elements and products as a string of coset representatives in the derived series, then S_ε can be computed in time polynomial in its size.

Proof. Solvable groups can be approached via Clifford theory, which controls the structure of representations of a group G when restricted to a normal subgroup. In fact, we require only a simple fact about this setting. Namely, if $H \lhd G$ and ρ is

an irrep of G, then either $\mathrm{Res}_H \rho$ contains only copies of the trivial representation so that $\rho(h) = \mathbb{1}_{\rho_d}$ for all $h \in H$, or $\mathrm{Res}_H \rho$ contains *no* copies of the trivial representation.

It is easy to see that the irreps ρ of G for which $\mathrm{Res}_H \rho$ is trivial are in one-to-one correspondence with irreps of the group G/H, and we will blur this distinction. With this perspective, it is natural to attempt to assemble an ε-biased set for G from S_H, an ε_H-biased set for H, and $S_{G/H}$, an $\varepsilon_{G/H}$-biased set for G/H. While $S_H \subset H \subset G$, there is—in general—no subgroup of G isomorphic to G/H, so it is not clear how to appropriately embed $S_{G/H}$ into G. Happily, we will see that reasonable bounds can be obtained even with an arbitrary embedding. In particular, we treat $S_{G/H}$ as a subset of G by lifting each element $x \in S_{G/H}$ to an arbitrary element $\hat{x} \in G$ lying in the H-coset associated with x.

If S_H and $S_{G/H}$ were the same size, and we could directly introduce an expander graph Γ on $S_H \times S_{G/H}$, then Lemma 2 could still be used to control the bias of $S = \{s\hat{t} \mid (s,t) \in \Gamma\}$. Specifically, consider a nontrivial representation ρ of G. If $\mathrm{Res}_H \rho$ is trivial, then analogous to (8) we have $\left\|\mathbb{E}_{s\in S} \rho(s)\right\| = \left\|\mathbb{E}_{s\in S_{G/H}} \rho(s)\right\| \leq \varepsilon_{G/H}$. On the other hand, if $\mathrm{Res}_H \rho$ restricts to H without any appearances of the trivial representation, then $\left\|\mathbb{E}_{h\in S_H} \rho(h)\right\| \leq \varepsilon_H$. In this case, the action of the elements of $S_{G/H}$ on ρ may be quite pathological, permuting and "twiddling" the H-irreps appearing in $\mathrm{Res}_H \rho$. However, as $\|\rho(s)\| = 1$ (by unitarity) for all $s \in S_{G/H}$, we can conclude from Lemma 2 that $\left\|\mathbb{E}_{s\in S} \rho(s)\right\| \leq \lambda(\Gamma) + \varepsilon_H$.

We recursively apply the construction outlined above, accounting for the "tiling error" of finding an appropriate expander. Specifically, let us inductively assume we have ϵ-biased sets S^+ on $G^+ = G/G^{(k)}$ and S^- on $G^- = G^{(k)}$ for $k = \lceil \ell/2 \rceil$, where ℓ is the derived length of G. Selecting an expander graph Γ of size at least $\alpha^{-1} \max(|S^-|, |S^+|)$ and $\lambda(\Gamma) \leq \alpha$, for an α to be determined, we tile each side of the graph with elements from S^- and S^+, completing them arbitrarily on the "uncovered elements." Since at most a fraction α of the elements on either side are uncovered, the average of a nontrivial representation over either side of the expander has operator norm no more than $\epsilon+\alpha$. Lemma 2 then implies that the bias of the set $S = \{st \mid (s,t) \in \Gamma\}$ is at most $\lambda(\Gamma) + (\epsilon + \alpha) \leq \epsilon + 2\alpha$. If we use the Ramanujan graphs of [9] described above, we can achieve degree $O(\alpha^{-2})$ and size $O(\alpha \max(|S^-|, |S^+|))$. Thus, each recursive step of this process scales the sizes of the sets by a factor $O(\alpha^{-3})$ and introduces additive error 2α. The number of levels of recursion is $\lceil \log_2 \ell \rceil$, so if we choose $\alpha < 1/(4\lceil \log \ell \rceil)$ then the total accumulated error is less than $1/2$.

Assuming that we have α-biased sets for each abelian factor A_i of size no more than s, this yields a $1/2$-biased set S for G of size $s\alpha^{-3\log_2 \ell} = s(\log \ell)^{O(\log \ell)}$. For constant p, there are α-biased sets for $\mathbb{Z}_p^n$ [2] of size $s = O(n/\alpha^3) = (\log |G|)(\log \ell)^{O(1)}$. Using the fact [8] that $\ell = O(\log \log |G|)$, the total size of S is $(\log |G|)(\log \ell)^{O(\log \ell)} = (\log |G|)(\log \log \log |G|)^{O(\log \log \log |G|)} = (\log |G|)^{1+o(1)}$. Finally, we amplify S to an ε-biased set S_ε for whatever ε we desire with Theorem 3, introducing a factor $O(\varepsilon^{-11})$.

Acknowledgments. We thank Amnon Ta-Shma, Emanuele Viola, and Avi Wigderson for helpful discussions. This work was supported by NSF grant CCF-1117426, CCF-1117427, and ARO contract W911NF-04-R-0009.

A Quadratic Forms Associated with Expander Graphs

Our goal is to establish the two generalized Rayleigh quotient bounds described in Lemmas 5 and 2. We begin with the following preparatory lemma.

Lemma 4. *Let $G = (U, V; E)$ be a (n, d, λ)-expander. Associate with each vertex $s \in U \cup V$ a vector $\mathbf{x^s}$ in $\mathbb{C}^d$ such that $\mathbb{E}_{u \in U} \mathbf{x^u} = 0$ and $\mathbb{E}_{v \in V} \mathbf{x^v} = 0$. Then $\left|\mathbb{E}_{(u,v)\in E} \langle \mathbf{x^u}, \mathbf{x^v} \rangle\right| \leq \lambda \, \mathbb{E}_s \|\mathbf{x^s}\|^2$.*

Proof. Let X denote the $2n \times d$ matrix whose entries are $X_{sk} = x^s_k$. Then the rows of X are the vectors $\mathbf{x}$; for an column index $k \in \{1, \ldots, d\}$, we let $\mathbf{y^k} \in \mathbb{C}^{2n}$ denote the vector associated with this column: $y^k_v = x^v_k$. Considering that $\sum_u \mathbf{x^u} = \sum_v \mathbf{x^v} = 0$, each $\mathbf{y^k}$ is orthogonal to both $\mathbf{y}^+$ and $\mathbf{y}^-$.

The expectation over a random edge (u, v) of $\langle \mathbf{x^u}, \mathbf{x^v} \rangle$ can be written

$$\begin{aligned}
\left| \mathop{\mathbb{E}}_{(u,v)\in E} \langle \mathbf{x^u}, \mathbf{x^v} \rangle \right| &= \left| \mathop{\mathbb{E}}_{(u,v)\in E} \sum_k X_{uk} X_{vk} \right| = \left| \sum_k \mathop{\mathbb{E}}_{(u,v)\in E} X_{uk} X_{vk} \right| \\
&= \left| \sum_k \frac{1}{nd} \sum_{(u,v)\in E} x^u_k x^v_k \right| = \left| \frac{1}{n} \sum_k \frac{1}{d} \sum_{(u,v)\in E} y^k_u y^k_v \right| \\
&= \frac{1}{2n} \left| \sum_k \left\langle \mathbf{y^k}, A\mathbf{y^k} \right\rangle \right| \leq \frac{1}{2n} \sum_k \left| \left\langle \mathbf{y^k}, A\mathbf{y^k} \right\rangle \right| \\
&\leq \frac{\lambda}{2n} \sum_k \|\mathbf{y^k}\|^2 = \frac{\lambda}{2n} \sum_s \|\mathbf{x^s}\|^2 = \lambda \mathop{\mathbb{E}}_s \|\mathbf{x^s}\|^2 .
\end{aligned}$$

Lemma 5. *Let $G = (U, V; E)$ be a (n, d, λ)-expander. Associate with each vertex $s \in U \cup V$ a vector $\mathbf{x^s}$ in $\mathbb{C}^d$ such that $\|\mathbb{E}_{u \in U} \mathbf{x^u}\| = \varepsilon_U$ and $\|\mathbb{E}_{v \in V} \mathbf{x^v}\| = \varepsilon_V$. Then $\left|\mathbb{E}_{(u,v)\in E} \langle \mathbf{x^u}, \mathbf{x^v} \rangle\right| \leq \lambda \left(\mathbb{E}_s \|\mathbf{x^s}\|^2 - \varepsilon_U^2/2 - \varepsilon_V^2/2\right) + \varepsilon_U \varepsilon_V$.*

Proof. Let $\mathbf{x}^U = \mathbb{E}_{u \in U} \mathbf{x^u}$ and $\mathbf{x}^V = \mathbb{E}_{v \in V} \mathbf{x^v}$. We have

$$\left| \mathop{\mathbb{E}}_{(u,v)\in E} \langle \mathbf{x^u}, \mathbf{x^v} \rangle \right| = \left| \mathop{\mathbb{E}}_{(u,v)\in E} \langle (\mathbf{x^u} - \mathbf{x^U}) + \mathbf{x^U}, (\mathbf{x^v} - \mathbf{x^V}) + \mathbf{x^V} \rangle \right| .$$

Expanding these inner products by linearity, and using the fact that the terms $\mathbb{E}_{(u,v)\in E} \langle \mathbf{x^U}, (\mathbf{x^v} - \mathbf{x^V}) \rangle$ and $\mathbb{E}_{(u,v)\in E} \langle (\mathbf{x^u} - \mathbf{x^U}), \mathbf{x^V} \rangle$ are zero, we conclude that

$$\left| \mathop{\mathbb{E}}_{(u,v)\in E} \langle \mathbf{x^u}, \mathbf{x^v} \rangle \right| \leq \left| \mathop{\mathbb{E}}_{(u,v)\in E} \langle (\mathbf{x^u} - \mathbf{x^U}), (\mathbf{x^v} - \mathbf{x^V}) \rangle \right| + \left| \langle \mathbf{x^U}, \mathbf{x^V} \rangle \right| .$$

Applying Lemma 4 to the the vectors $\mathbf{x^u} - \mathbf{x^U}$ and $\mathbf{x^v} - \mathbf{x^V}$, we conclude that $\left|\mathbb{E}_{(u,v)\in E} \langle (\mathbf{x^u} - \mathbf{x^U}), (\mathbf{x^v} - \mathbf{x^V}) \rangle\right| \leq (\lambda/2n) \left(\sum_u \|\mathbf{x^u} - \mathbf{x^U}\|^2 + \sum_v \|\mathbf{x^v} - \mathbf{x^V}\|^2\right)$.

The summation $\sum_u \|\mathbf{x}^{\mathbf{u}} - \mathbf{x}^{\mathbf{U}}\|^2$ can be expanded

$$\sum_u \left\langle \mathbf{x}^{\mathbf{u}} - \mathbf{x}^{\mathbf{U}}, \mathbf{x}^{\mathbf{u}} - \mathbf{x}^{\mathbf{U}} \right\rangle = \sum_u \|\mathbf{x}^{\mathbf{u}}\|^2 - n\|\mathbf{x}^{\mathbf{U}}\|^2 = \sum_u \|\mathbf{x}^{\mathbf{u}}\|^2 - n\varepsilon_U^2$$

since $\sum_u \left\langle \mathbf{x}^{\mathbf{u}}, \mathbf{x}^{\mathbf{U}} \right\rangle = n\|\mathbf{x}^{\mathbf{U}}\|^2$. Therefore,

$$\begin{aligned}\left| \mathop{\mathbb{E}}_{(u,v)\in E} \langle (\mathbf{x}^{\mathbf{u}} - \mathbf{x}^{\mathbf{U}}), (\mathbf{x}^{\mathbf{v}} - \mathbf{x}^{\mathbf{V}}) \rangle \right| &\le \frac{\lambda}{2n} \left(\sum_u \|\mathbf{x}^{\mathbf{u}}\|^2 - n\varepsilon_U^2 + \sum_v \|\mathbf{x}^{\mathbf{v}}\|^2 - n\varepsilon_V^2 \right) \\ &\le \lambda \left(\mathop{\mathbb{E}}_s \|\mathbf{x}^{\mathbf{s}}\|^2 - \frac{\varepsilon_U^2}{2} - \frac{\varepsilon_V^2}{2} \right) .\end{aligned}$$

By Cauchy-Schwarz, we have $|\langle \mathbf{x}^{\mathbf{U}}, \mathbf{x}^{\mathbf{V}} \rangle| \le \varepsilon_U \varepsilon_V$. In total, then,

$$\left| \mathop{\mathbb{E}}_{(u,v)\in E} \langle \mathbf{x}^{\mathbf{u}}, \mathbf{x}^{\mathbf{v}} \rangle \right| \le \lambda \left(\mathop{\mathbb{E}}_s \|\mathbf{x}^{\mathbf{s}}\|^2 - \frac{\varepsilon_U^2}{2} - \frac{\varepsilon_V^2}{2} \right) + \varepsilon_U \varepsilon_V ,$$

as desired.

We now return to the proof of Lemma 2.

Proof (of Lemma 2). Let X denote the linear operator $\mathbb{E}_{(u,v)\in E} X_u X_v$. Writing $\|X\| = \max_{\|\mathbf{x}\|=\|\mathbf{y}\|=1} |\langle \mathbf{x}, X\mathbf{y} \rangle|$. we observe that $\langle \mathbf{x}, X\mathbf{y} \rangle = \left\langle \mathbf{x}, \mathbb{E}_{(u,v)\in E} X_u X_v \mathbf{y} \right\rangle = \mathbb{E}_{(u,v)\in E} \langle X_u^\dagger \mathbf{x}, X_v \mathbf{y} \rangle$. Considering the bounds on $\mathbb{E}_u X_u$ and $\mathbb{E}_v X_v$, it follows that $\| \mathbb{E}_u X_u^\dagger \mathbf{x} \| \le \varepsilon_U$ and $\| \mathbb{E}_v X_v \mathbf{y} \| \le \varepsilon_V$; applying Lemma 5 with the vector family $\mathbf{x}^{\mathbf{u}} = X_u^\dagger \mathbf{x}$ and $\mathbf{x}^{\mathbf{v}} = X_v \mathbf{y}$ we conclude that

$$|\langle \mathbf{x}, X\mathbf{y} \rangle| \le \max_{\substack{\delta_U \le \varepsilon_U \\ \delta_V \le \varepsilon_V}} \lambda \left(\mathop{\mathbb{E}}_s \|\mathbf{x}^{\mathbf{s}}\|^2 - \frac{\delta_U^2}{2} - \frac{\delta_V^2}{2} \right) + \delta_U \delta_V \le \lambda + (1-\lambda)\varepsilon_U \varepsilon_V$$

as $\delta_U^2 + \delta_V^2 \ge 2\delta_U \delta_V$.

B A Tail Bound for Products of Operator-Valued Random Variables

Recall Azuma's inequality for supermartingales:

Theorem 7 (Azuma's inequality). *Let $X_0, \ldots, X_T$ be a family of real-valued random variables for which $|X_i - X_{i-1}| \le \alpha_i$ and $\mathbb{E}[X_i \mid X_1, \ldots, X_{i-1}] \le X_{i-1}$. Then $\Pr[X_T - X_0 \ge \lambda] \le \exp\left(-\lambda^2/(2\alpha)\right)$, where $\alpha = \sum_i \alpha_i$.*

Corollary 1. *Let $X_0, \ldots, X_T$ be a family of real-valued random variables for which $X_{i-1} - \alpha_i \le X_i \le X_{i-1}$ and $\mathbb{E}[X_i \mid X_1, \ldots, X_{i-1}] \le X_{i-1} - \varepsilon_i$ for some $\varepsilon_i \le \alpha_i$. Then $\Pr[X_T - X_0 \ge -\sum_i \varepsilon_i + \lambda] \le \exp\left(-\lambda^2/(2\alpha)\right)$, where $\alpha = \sum_i \alpha_i$.*

Proof. Apply Azuma's inequality to the random variables $\tilde{X}_t = X_t + \sum_i^t \varepsilon_i$.

With these in place, we return to the proof of Theorem 2.

Proof (of Theorem 2). We begin by considering the behavior of the operator $P_k \cdots P_1$ on a particular vector $\mathbf{v}$. To complete the proof we will select an orthonormal basis $\mathcal{B}$ of H. The operator norm is bounded above by the Frobenius norm,

$$\|P_k \cdots P_1\| \leq \|P_k \cdots P_1\|_F = \sqrt{\sum_{\mathbf{b}\in\mathcal{B}} \|P_k \dots P_1 \mathbf{b}\|^2} \leq \sqrt{\dim H} \cdot \max_{\mathbf{b}\in\mathcal{B}} \|P_k \dots P_1 \mathbf{b}\| . \tag{11}$$

Now fix a unit-length vector $\mathbf{v} \in H$ and consider the random variables $\mathbf{v_0} = \mathbf{v}$, $\mathbf{v_1} = P_1\mathbf{v}$, $\mathbf{v_2} = P_2 P_1 \mathbf{v}$, ..., and

$$\ell_i = \begin{cases} \|\mathbf{v_i}\|/\|\mathbf{v_{i-1}}\| & \text{if } \mathbf{v_{i-1}} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Our goal is to establish strong tail bounds on the random variable $\|\mathbf{v_k}\| = \ell_k \ell_{k-1} \dots \ell_1$. Recalling that $\|\mathbb{E}[P_i]\| \leq 1 - \varepsilon$ and that the P_i are independent we have

$$\mathbb{E}\left[\ell_i \mid P_1, \dots, P_{i-1}\right] \leq 1 - \varepsilon , \tag{12}$$

and we proceed to apply a martingale tail bound.

It will be more convenient to work with log-bounded random variables, so we define $m_i = \max(\ell_i, 1/2)$ and observe that $\|\mathbf{v_k}\| \leq m_k m_{k-1} \dots m_1$ and $\ln \|\mathbf{v_k}\| \leq \sum_i \ln m_i$. Considering that $\max(x, 1/2) \leq (1+x)/2$ for $x \in [0,1]$ we conclude from equation (12) above that $\mathbb{E}\left[m_i \mid P_1, \dots, P_{i-1}\right] \leq 1 - \varepsilon/2$. Since $1/2 \leq m_i \leq 1$ and $\ln m \leq m - 1$ for $m > 0$, we have $\mathbb{E}\left[\ln m_i \mid P_1, \dots, P_{i-1}\right] \leq -\varepsilon/2$.

Applying Azuma's inequality (specifically, Corollary 1 above) to the random variables $M_t = \sum_{i=1}^t \ln m_i$, we conclude that

$$\Pr\left[M_k \geq -\frac{k\varepsilon}{2} + \Delta\right] = \Pr\left[\sum_i \ln m_i \geq -\frac{k\varepsilon}{2} + \Delta\right] \leq \exp\left(-\frac{\Delta^2}{2k \ln 2}\right)$$

and hence $\Pr\left[\|P_k \cdots P_1 \mathbf{v}\| \geq \exp\left(-\frac{k\varepsilon}{2} + \Delta\right)\right] \leq \exp\left(-\Delta^2/(2k \ln 2)\right)$. Applying the above inequality to an orthonormal basis $\mathbf{b_1}, \dots, \mathbf{b_n}$ of H, we find that

$$\Pr\left[\exists i : \left\|P_k \cdots P_1 \mathbf{b_i}\right\|_2 \geq \exp\left(-\frac{k\varepsilon}{2} + \Delta\right)\right] \leq \dim H \cdot \exp\left(-\frac{\Delta^2}{2k \ln 2}\right)$$

by the union bound. Applying (11) then gives

$$\Pr\left[\left\|P_k \cdots P_1\right\| \geq \sqrt{\dim H} \exp\left(-\frac{k\varepsilon}{2} + \Delta\right)\right] \leq \dim H \cdot \exp\left(-\frac{\Delta^2}{2k \ln 2}\right) .$$

References

[1] Alon, N., Roichman, Y.: Random Cayley graphs and expanders. Random Structures and Algorithms 5(2), 271–284 (1994)

[2] Alon, N., Bruck, J., Naor, J., Naor, M., Roth, R.M.: Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. IEEE Transactions on Information Theory 38(2), 509–516 (1992)

[3] Alon, N., Goldreich, O., Håstad, J., Peralta, R.: Simple construction of almost k-wise independent random variables. Random Struct. Algorithms 3(3), 289–304 (1992)

[4] Ben-Aroya, A., Ta-Shma, A.: Constructing small-bias sets from algebraic-geometric codes. Theory of Computing 9(5), 253–272 (2013)

[5] Bogdanov, A., Viola, E.: Pseudorandom bits for polynomials. SIAM Journal on Computing 39(6), 2464–2486 (2010)

[6] De, A.: Pseudorandomness for permutation and regular branching programs. In: 2011 IEEE 26th Annual Conference on Computational Complexity (CCC), pp. 221–231 (2011)

[7] Friedl, K., Ivanyos, G., Magniez, F., Santha, M., Sen, P.: Hidden translation and orbit coset in quantum computing. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing, pp. 1–9 (2003)

[8] Glasby, S.P.: The composition and derived lengths of a soluble group. J. Algebra 120, 406–413 (1989)

[9] Lubotzky, A., Phillips, R., Sarnak, P.: Ramanujan graphs. Combinatorica 8(3), 261–277 (1988)

[10] Meka, R., Zuckerman, D.: Small-bias spaces for group products. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX and RANDOM 2009. LNCS, vol. 5687, pp. 658–672. Springer, Heidelberg (2009)

[11] Moore, C., Rockmore, D., Russell, A.: Generic quantum fourier transforms. ACM Transactions on Algorithms 2(4), 707–723 (2006)

[12] Morgenstern, M.: Existence and explicit constructions of q + 1 regular Ramanujan graphs for every prime power q. Journal of Combinatorial Theory, Series B 62(1), 44–62 (1994)

[13] Naor, J., Naor, M.: Small-bias probability spaces: Efficient constructions and applications. SIAM Journal on Computing 22(4), 838–856 (1993)

[14] Rosser, J.B., Schoenfeld, L.: Approximate formulas for some functions of prime numbers. Illinois Journal of Mathematics 6, 64–94 (1962)

[15] Rozenman, E., Vadhan, S.: Derandomized squaring of graphs. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX and RANDOM 2005. LNCS, vol. 3624, pp. 436–447. Springer, Heidelberg (2005)

[16] Wigderson, A., Xiao, D.: Derandomizing the Ahlswede-Winter matrix-valued Chernoff bound using pessimistic estimators, and applications. Theory of Computing 4(3), 53–76 (2008)

What You Can Do with Coordinated Samples

Edith Cohen[1,2] and Haim Kaplan[2]

[1] Microsoft Research, SVC, USA
edith@cohenwang.com
[2] The Blavatnik School of Computer Science, Tel Aviv University, Israel
haimk@cs.tau.ac.il

Abstract. Sample coordination, where similar instances have similar samples, was proposed by statisticians four decades ago as a way to maximize overlap in repeated surveys. Coordinated sampling had been since used for summarizing massive data sets.

The usefulness of a sampling scheme hinges on the scope and accuracy within which queries posed over the original data can be answered from the sample. We aim here to gain a fundamental understanding of the limits and potential of coordination. Our main result is a precise characterization, in terms of simple properties of the estimated function, of queries for which estimators with desirable properties exist. We consider unbiasedness, nonnegativity, finite variance, and bounded estimates.

Since generally a single estimator can not be optimal (minimize variance simultaneously) for all data, we propose *variance competitiveness*, which means that the expectation of the square on any data is not too far from the minimum one possible for the data. Surprisingly perhaps, we show how to construct, for any function for which an unbiased nonnegative estimator exists, a variance competitive estimator.

1 Introduction

Many data sources, including IP or Web traffic logs from different time periods or locations, measurement data, snapshots of data depositories that evolve over time, and document/feature and market-basket data, can be viewed as a collection of *instances*, where each instance is an assignment of numeric values from some set V to a set of items (the set of items is the same for different instances but the value of each item changes).

When the data is too massive to manipulate or even store in full, it is useful to obtain and work with a random sample of each instance. Two common sampling schemes are Poisson sampling (each item is sampled independently with probability that depends only on its value) and bottom-k (order) sampling. The samples are efficient to compute, also when instances are presented as streams or are distributed across multiple servers. It is convenient to specify these sampling schemes through a *rank function*, $r : [0,1] \times V \to \mathbb{R}$, which maps seed-value pairs to a number $r(u,v)$ that is non-increasing with u and non-decreasing with v. For each item h we draw a *seed* $u(h) \sim U[0,1]$ uniformly at random and compute the rank value $r(u(h), v(h))$, where $v(h)$ is the value of h. With Poisson sampling, item h is sampled $\iff r(u(h), v(h)) \geq T(h)$, where $T(h)$

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 452–467, 2013.

are fixed thresholds, whereas a bottom-k sample includes the k keys with highest ranks. Poisson PPS samples (Probability Proportional to Size [23], where each item is included with probability proportion to its value) are obtained using the rank function $r(u, v) = v/u$ and a fixed $T(h)$ across items. Priority (sequential Poisson) samples [29,17,34] are bottom-k samples utilizing the PPS ranks $r(u, v) = v/u$ and successive weighted sampling without replacement [31,18,8] corresponds to bottom-k samples with the rank function $r(u, v) = -v/\ln(u)$.

Samples of different instances are *coordinated* when the set of random seeds $u(h)$ is shared across instances. Scalable sharing of seeds when instances are dispersed in time or location is facilitated through random hash functions $u(h) \leftarrow H(h)$, where the only requirement for our purposes is uniformity and pairwise independence. Figure 1 contains an example data set of two instances and the PPS sampling probabilities of each item in each instance, and illustrates how to coordinate the samples. Note that the sample of one instance does not depend on values assumed in other instances, which is important for scalable deployment.

Why coordinate samples? Sample coordination was proposed in 1972 by Brewer, Early, and Joice [2], as a method to maximize overlap and therefore minimize overhead in repeated surveys [33,30,32]: The values of items change, and therefore there is a new set of PPS sampling probabilities. With coordination, the sample of the new instance is as similar as possible to the previous sample, and therefore the number of items that need to be surveyed again is minimized. Coordination was subsequently used to facilitate efficient processing of large data sets. Coordinated samples of instances are used as synopses which facilitate efficient estimation of multi-instance functions such as distinct counts (cardinality of set unions), sum of maxima, and similarity [4,3,6,15,28,19,20,5,16,9,1,22,10,14]. Estimates obtained over coordinated samples are much more accurate than possible with independent samples. Used this way, coordinated sampling can be casted as a form of Locality Sensitive Hashing (LSH) [26,21,25]. Lastly, coordinated samples can sometimes be obtained much more efficiently than independent samples. One example is computing samples of the d-neighborhoods of all nodes in a graph [6,7,28,8,9]. Similarity queries between neighborhoods are useful in the analysis of massive graph datasets such as social networks or Web graphs.

Our aim here is to study the potential and limitations of estimating multi-instance functions from coordinated samples of instances. The same set of samples can be used to estimate multiple queries. We therefore do not aim for a sampling scheme optimized for a particular query (although some of our results can be applied this way), but rather, to optimize the estimator given the sampling scheme and query.

Sum Aggregates: Most queries in the above examples can be casted as *sum aggregates* over selected items h of a *basic function* $f(\boldsymbol{v})$ applied to the item weight tuple in different instances $\boldsymbol{v}(h) = (v_1(h), v_2(h), \cdots)$. In particular, distinct count (set union) is a sum aggregate of $OR(\boldsymbol{v})$, max-sum aggregates $\max(\boldsymbol{v}) = \max_i v_i$, min-sum aggregates $\min(\boldsymbol{v}) = \min_i v_i$, and L_p^p (pth power of L$_p$-difference) is the sum aggregate of the exponentiated range function $\text{RG}_p(\boldsymbol{v}) = |\max(\boldsymbol{v}) - \min(\boldsymbol{v})|^p$. In our example of Figure 1, L_2^2 of items $[4]$ is $(1-3)^2+(2-0)^2+(4-1)^2+(1-0)^2 = 18$, and is computed by summing the basic function $\text{RG}_2(v_1, v_2) = (v_1 - v_2)^2$ over these items. The L_1 of items $\{1, 3\}$ is $|1 - 3| + |4 - 1| = 5$, using the basic function $\text{RG}(v_1, v_2) = |v_1 - v_2|$,

items:	1	2	3	4	5	6	7	8
Instance1:	1	0	4	1	0	2	3	1
Instance2:	3	2	1	0	2	3	1	0
PPS sampling probabilities for T=4 (sample of expected size 3):								
Instance1:	0.25	0.00	1.00	0.25	0.00	0.50	0.75	0.25
Instance2:	0.75	0.50	0.25	0.00	0.50	0.75	0.25	0.00

Fig. 1. Two instances with 8 items and respective PPS sampling probabilities for threshold value 4, so item with value v is sampled with probability $\min\{1, v/4\}$. To obtain two coordinated PPS samples of the instances, we associate an independent $u(i) \sim U[0,1]$ with each item $i \in [8]$. We then sample $i \in [8]$ in instance $h \in [2]$ if and only if $u(i) \leq v_h(i)/4$, where $v_h(i)$ is the value of i in instance h. When coordinating the samples this way, we make them as similar as possible. In the example, item 1 will always (for any drawing of seeds) be sampled in instance 2 if it is sampled in instance 1 and vice versa for item 7.

and the max-sum of items $\{6,7,8\}$ is $\max\{2,3\}+\max\{3,1\}+\max\{1,0\} = 7$, which uses the basic function $\max\{v_1, v_2\}$. Moreover, other queries including Jaccard similarity and L_p difference which are not sum aggregates can be approximated well by sum aggregates.

Sum Estimators – One Tuple at a Time: To estimate a sum aggregate, we can use a linear estimator which is the sum of *single-tuple estimators*, estimating the basic function $f(\boldsymbol{v}(h))$ for each selected item h. We refer to such an estimator as a *sum estimator*. When the single-tuple estimators are unbiased, from linearity of expectation, so is the sum estimate. When the single-tuple estimators are unbiased and sampling of different tuples is pairwise independent (respectively, negatively correlated, as with bottom-k sampling), the variance of the sum is (resp., at most) the sum of variances of the single-tuple estimators. Therefore, the relative error of the sum estimator decreases with the number of selected items we aggregate over. We emphasize that unbiasedness of the single-tuple estimators (together with pairwise independence or negative correlations between tuples) is critical for good estimates of the sum aggregate since a variance component that is due to bias "adds up" with aggregation whereas otherwise the relative error "cancels out" with aggregation. The Horvitz-Thompson (HT) estimator [24] is a classic sum estimator which is unbiased and nonnegative. To estimate $f(v)$, the HT estimator outputs 0 when the value is not sampled and the inverse-probability estimate $f(v)/p$ when the value is sampled, where p is the sampling probability. The HT estimator is applicable to some multi-instance functions [14].

From here on, we restrict our attention to estimating single-tuple functions $f(\boldsymbol{v}) \geq 0$ where each entry of $\boldsymbol{v}$ is Poisson sampled and focus on unbiased and nonnegative estimators for $f(\boldsymbol{v})$. Our model is provided in detail in Section 2.

The Challenge We Address: Throughout the 40 year period in which coordination was used, estimators were developed in an ad-hoc manner, lacking a fundamental understanding of the potential and limits of the approach. Prior work was mostly based on adaptations of the HT estimator for multiple instances. The HT estimator is applicable provided that for any $\boldsymbol{v}$ where $f(\boldsymbol{v}) > 0$, there is a positive probability for an outcome that both reveals $f(\boldsymbol{v})$ and allows us to determine a probability p for such an outcome. These conditions are satisfied by some basic functions including $\max(\boldsymbol{v})$ and $\min(\boldsymbol{v})$.

There are functions, however, for which the HT estimator is not applicable, but nonetheless, for which nonnegative and unbiased estimators exist. Moreover, the HT estimator may not be optimal even when it is applicable.

As a particular example, the only L_p difference for which a "satisfactory" estimator was known was the L_1 difference [14]. Stated in terms of single-tuple estimators, prior to our work, there was no unbiased and nonnegative estimator known for $\mathrm{RG}_p(\boldsymbol{v}) = |\max(\boldsymbol{v}) - \min(\boldsymbol{v})|^p$ for any $p \neq 1$. For $p = 1$, the known estimator used the relation $\mathrm{RG}(\boldsymbol{v}) = \max(\boldsymbol{v}) - \min(\boldsymbol{v})$, separately estimating the maximum and the minimum and showing that when samples are coordinated then the estimate for the maximum is always at least as large as the one for the minimum and therefore the difference of the estimates is never negative. But even for this $\mathrm{RG}(\boldsymbol{v})$ estimator, there was no understanding whether it is "optimal" and more so, what optimality even means in this context. Moreover, the ad hoc construction of this estimator does not extend even to slight variations, like $\max\{v_1 - v_2, 0\}$, which sum-aggregates to the natural "one sided" L_1 difference.

Contributions Highlights

Characterization: We provide a complete characterization, in terms of simple properties of the function f and the sampling scheme parameters, of when estimators with the following combinations of properties exist for f:

- unbiasedness and nonnegativity.
- unbiasedness, nonnegativity, and *finite variances*, which means that for all $\boldsymbol{v}$, the variance given data $\boldsymbol{v}$ is finite.
- unbiasedness, nonnegativity, and *bounded estimates*, which means that for each $\boldsymbol{v}$, there is an upper bound on all estimates that can be obtained when the data is $\boldsymbol{v}$. Bounded estimates implies finite variances, but not vice versa.

The J Estimator: Our characterization utilizes a construction of an estimator, which we call *the J estimator*, which we show has the following properties: The J estimator is unbiased and nonnegative if and only if an unbiased nonnegative estimator exists for f. The J estimator has a finite variance for data $\boldsymbol{v}$ or is bounded for data $\boldsymbol{v}$ if and only if an (nonnegative unbiased) estimator with the respective property for $\boldsymbol{v}$ exists.

Variance Competitiveness: Generally, there may not be a single (unbiased, nonnegative, linear) estimator with minimum variance on all data vectors [27]. We are therefore aiming for a notion of *variance competitiveness*, which means that for *any* data vector, the variance of our estimator is not "too far" from the minimum variance possible for that vector by a nonnegative unbiased estimator. More precisely, an estimator is c-competitive if for all data $\boldsymbol{v}$, the expectation of its square is within a factor of c from the minimum possible for $\boldsymbol{v}$ by an estimator that is unbiased and nonnegative on all data.

$\boldsymbol{v}$-Optimality: To study competitiveness, we need to compare the variance on each data vector to the minimum possible, and to do so, we need to be able to express the "best possible" estimates. We say that an estimator is *$\boldsymbol{v}$-optimal* if amongst all estimators that are unbiased and nonnegative on all data, it minimizes variance for the data $\boldsymbol{v}$. We express the $\boldsymbol{v}$-optimal estimates, which are the values a $\boldsymbol{v}$-optimal estimator assumes on outcomes that are consistent with data $\boldsymbol{v}$, and the respective $\boldsymbol{v}$-optimal variance.

We show that the $\boldsymbol{v}$-optimal estimates are uniquely defined (almost everywhere). The $\boldsymbol{v}$-optimal estimates, however, are inconsistent for different data since as we mentioned earlier, it is not generally possible to obtain a single unbiased nonnegative estimator that minimizes variance for all data vectors. They do allow us, however, to analyse the competitiveness of estimators, and in particular, that of the J estimator.

Competitiveness of the J Estimator: We show that the J estimator is competitive. In particular, this shows the powerful and perhaps surprising result that whenever for any particular data vector $\boldsymbol{v}$ there exists an estimator with finite variance that is nonnegative and unbiased on all data, then there is a *single* estimator, that for all data, has expectation of the square that is $O(1)$ of the minimum possible.

Practical Implications: We demonstrate some of the practical significance of our work in [12,13], where we derive and apply L_p difference estimators for the exponentiated range functions RG_p ($p > 0$), and experimentally study the performance of the L_1 and L_2 estimators over PPS (and priority) samples of various data sets.

The study demonstrates accurate estimates even when a small fraction of the data set is sampled. To the best of our knowledge, prior to our work, there was no good estimator for L_p differences over coordinated samples for any $p \neq 1$ and only a weaker estimator was known for $p = 1$ [14]. The competitive ratios of the estimators studied in [13] are 2 for the L_1 estimator and 2.5 for the L_2 estimator.

2 Coordinated Sampling Model

The data is a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_r) \in \mathbf{V} = V^r$, where $\mathbf{V} \subseteq \mathbb{R}^r_{\geq 0}$. Using the terminology in the introduction, we are now looking at a single item (single-tuple $\boldsymbol{v}$) and the value v_i of the ith entry is the value of the item in instance i. The data is sampled through a *sampling scheme* specified by non-decreasing continuous functions $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_r)$ on $[0, 1]$ with range containing $(\min V, \max V)$. The outcome $S \equiv S(u, \boldsymbol{v})$ is the output of the sampling scheme and is a function of a random seed $u \in U[0, 1]$ and the data $\boldsymbol{v}$. We treat the outcome as a set where the ith entry is included in S if and only if v_i is at least $\tau_i(u)$:

$$i \in S \iff v_i \geq \tau_i(u) \, .$$

Sampling is PPS if $\tau_i(u)$ are linear functions: there is a fixed vector $\boldsymbol{\tau^*}$ such that $\tau_i(u) \equiv u\tau_i^*$, in which case entry i is included with probability $\min\{1, v_i/\tau_i^*\}$. Our use of the term PPS refers to sampling of each instance i using threshold τ_i^*, and the "projected" sampling scheme on the ith entry of our tuple.

Observe that our model assumes *weighted* sampling, where the probability that an entry is sampled depends (and is non-decreasing) with its value. Transiting briefly back to sampling of instances, weighted sampling results in more accurate estimation of quantities (such as averages of sums) where larger values contribute more. It is also important for boolean domains ($V = \{0, 1\}$) when most items have 0 values and in this case, enables us to sample only "active" items.

We assume that the seed u and the functions $\boldsymbol{\tau}$ are available to the estimator, and in particular, treat the seed as provided with the outcome. When an entry is sampled,

we know its value and also can compute the probability that it is sampled. When an entry is not sampled, we know that its value is at most $\tau_i(u)$ and we can compute this upper bound from the seed u and the function τ_i. Putting this information together, for each outcome $S(u, \boldsymbol{v})$, we can define the set $V^*(S)$ of all data vectors consistent with the outcome. This set captures all the information we can glean from the sample on the data.

$$V^*(S) \equiv V^*(u, \boldsymbol{v}) = \{\boldsymbol{z} \mid \forall i \in [r], (i \in S \wedge z_i = v_i) \vee (i \notin S \wedge z_i < \tau_i(u))\} .$$

Structure of the Set of Outcomes. From the outcome, which is the set of sampled entries and the seed ρ, we can determine $V^*(u, \boldsymbol{v})$ also for all $u \geq \rho$. We also have that for all $u \geq \rho$ and $\boldsymbol{z} \in V^*(\rho, v)$, $V^*(u, \boldsymbol{z}) = V^*(u, \boldsymbol{v})$. Fixing $\boldsymbol{v}$, the sets $V^*(u, \boldsymbol{v})$ are non-decreasing with u and the set S of sampled entries is non-increasing, meaning that $V^*(u, \boldsymbol{v}) \subset V^*(\rho, \boldsymbol{v})$ and $S(u, \boldsymbol{v}) \supset S(\rho, \boldsymbol{v})$ when $u < \rho$.

The containment order of the sets $V^*(S)$ is a tree-like partial order on outcomes. For two outcomes, the sets $V^*(S)$ are either disjoint, and unrelated in the containment order, or one is fully contained in another, and succeeds it in the containment order. The outcome $S(u, \boldsymbol{v})$ precedes $S(\rho, \boldsymbol{v})$ in the containment order if and only if $u > \rho$. When V is finite, the containment order is a tree order, as shown in Figure 2.

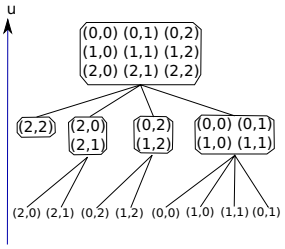


Fig. 2. Illustration of the containment order on all possible outcomes $V^*(S)$. Example has data vectors $V = \{0, 1, 2\} \times \{0, 1, 2\}$ and seed mappings $\tau_1 = \tau_2 \equiv \tau$. The root of the tree corresponds to outcomes with $u \in (\tau^{-1}(2), 1]$. In this case, the outcome reveals no information on the data and $V^*(S)$ contains all vectors in V. When $u \in (\tau^{-1}(1), \tau^{-1}(2)]$ the outcome identifies entries in the data that are equal to "2". When $u \in (0, \tau^{-1}(1)]$, the outcome reveals the data vector.

When V is an interval of the nonnegative reals, then for $\boldsymbol{z}$ and $\boldsymbol{v}$, the set of all u such that $\boldsymbol{z} \in S(u, \boldsymbol{v})$, if not empty, is a suffix of $(0, 1]$ that is open to the left.

Lemma 1

$$\forall \rho \in (0, 1] \, \forall \boldsymbol{v}$$
$$\boldsymbol{z} \in V^*(\rho, \boldsymbol{v}) \implies \exists \epsilon > 0, \, \forall x \in (\rho - \epsilon, 1], \, \boldsymbol{z} \in V^*(x, \boldsymbol{v})$$

Proof. Correctness for all $x \in [\rho, 1]$ follows from the structure of the set of outcomes: Since $V^*(x, \boldsymbol{v}) \supset V^*(\rho, \boldsymbol{v})$ for all $x \geq \rho$ then $\boldsymbol{z} \in V^*(\rho, \boldsymbol{v}) \implies \boldsymbol{z} \in V^*(x, \boldsymbol{v})$.

Consider now the set S of entries that satisfy $v_i \geq \tau_i(\rho)$. Since $\boldsymbol{z} \in V^*(\rho, \boldsymbol{v})$, we have $\forall i \in S, z_i = v_i$ and $\forall i \notin S, \, \max\{z_i, v_i\} < \tau_i(\rho)$. Since τ_i is continuous and

monotone, for all $i \notin S$, there must be an $\epsilon_i > 0$ such that $\tau_i(\rho - \epsilon_i) > \max\{z_i, v_i\}$. We now take $\epsilon = \min_{i \notin S} \epsilon_i$ to conclude the proof. □

3 Estimators and Properties

Let $f : \mathbf{V}$ be a function mapping $\mathbf{V}$ to the nonnegative reals. An *estimator* $\hat{f}$ of f is a numeric function applied to the outcome. We use the notation $\hat{f}(u, \boldsymbol{v}) \equiv \hat{f}(S(u, \boldsymbol{v}))$. On continuous domains, an estimator must be (Lebesgue) integrable. An estimator is *fully specified* for $\boldsymbol{v}$ if specified on a set of outcomes that have probability 1 given data $\boldsymbol{v}$. Two estimators $\hat{f}_1$ and $\hat{f}_2$ are *equivalent* if for all data $\boldsymbol{v}$, $\hat{f}_1(u, \boldsymbol{v}) = \hat{f}_2(u, \boldsymbol{v})$ with probability 1.

An estimator $\hat{f}$ is *nonnegative* if $\forall S,\ \hat{f}(S) \geq 0$ and is *unbiased* if $\forall \boldsymbol{v},\ \mathsf{E}[\hat{f}|\boldsymbol{v}] = f(\boldsymbol{v})$. An estimator has *finite variance* on $\boldsymbol{v}$ if $\int_0^1 \hat{f}(u, \boldsymbol{v})^2 du < \infty$ (the expectation of the square is finite) and is *bounded* on $\boldsymbol{v}$ if $\sup_{u \in (0,1]} \hat{f}(u, \boldsymbol{v}) < \infty$. If a nonnegative estimator is bounded on $\boldsymbol{v}$, it also has finite variance for $\boldsymbol{v}$. We say that an estimator is bounded or has finite variances if the respective property holds for all $\boldsymbol{v} \in \mathbf{V}$.

$\boldsymbol{v}$-Optimality. We say that an unbiased and nonnegative estimator is *$\boldsymbol{v}$-optimal*, that is, optimal with respect to a data vector $\boldsymbol{v}$, if it has minimum variance for $\boldsymbol{v}$. We refer to the estimates that a $\boldsymbol{v}$-optimal estimator assumes on outcomes consistent on data $\boldsymbol{v}$ as the *$\boldsymbol{v}$-optimal estimates* and to the minimum variance attainable for $\boldsymbol{v}$ as the *$\boldsymbol{v}$-optimal variance*.

Variance Competitiveness. An estimator $\hat{f}$ is *c-competitive* if

$$\forall \boldsymbol{v},\ \int_0^1 \left(\hat{f}(u, \boldsymbol{v})\right)^2 du \leq c \inf_{\hat{f}'} \int_0^1 \left(\hat{f}'(u, \boldsymbol{v})\right)^2 du,$$

where the infimum is over all unbiased nonnegative estimators $\hat{f}'$ of f. For any unbiased estimator, the expectation of the square is closely related to the variance:

$$\textsc{var}[\hat{f}|\boldsymbol{v}] = \int_0^1 (\hat{f}(u, \boldsymbol{v}) - f(\boldsymbol{v}))^2 du = \int_0^1 \hat{f}(u, \boldsymbol{v})^2 du - f(\boldsymbol{v})^2 \tag{1}$$

When minimizing the expectation of the square, we also minimize the variance. Moreover, c-competitiveness means that

$$\forall \boldsymbol{v},\ \textsc{var}[\hat{f}|\boldsymbol{v}] \leq c \inf_{\hat{f}'} \textsc{var}[\hat{f}'|\boldsymbol{v}] + (c-1) f(\boldsymbol{v})^2 \tag{2}$$

for all data vectors $\boldsymbol{v}$ for which a nonnegative unbiased estimator with finite variance on $\boldsymbol{v}$ exists, the variance of the estimator is at most c times the $\boldsymbol{v}$-optimal variance plus an additive term of $(c-1)$ times $f(\boldsymbol{v})^2$.

An important remark is due here. In the typical scenario, discussed in the introduction, the sample is likely to provide little or no information on $f(\boldsymbol{v})$, the variance is $\Omega(f(\boldsymbol{v})^2)$, and hence competitiveness as we defined it in terms of the expectation of the square translates to competitiveness of the variance. Otherwise, when for some data

in the domain the sample is likely to reveal the value, it is not possible to obtain a universal competitiveness result in terms of variance. (One such example is RG on PPS samples, looking at data tuples where the maximum has sampling probability 1.) Interestingly, for RG_2 it is possible to get a bounded ratio in terms of variance. More details are in the companion experimental paper [13].

4 The Lower Bound Function and Its Lower Hull

For a function f, we define the respective *lower bound function* $\underline{f}$ and the lower hull function H_f. We then characterize, in terms of properties of $\underline{f}$ and H_f when nonnegative unbiased estimators exists for f and when such estimators exist that also have finite variances or are bounded.

The lower bound function $\underline{f}(S)$: For $Z \subset \mathbf{V}$, we define $\underline{f}(Z) = \inf\{f(v) \mid v \in Z\}$ to be the tightest lower bound on the values of f on Z. We use the notation $\underline{f}(S) \equiv \underline{f}(V^*(S))$, $\underline{f}(\rho, \boldsymbol{v}) \equiv \underline{f}(V^*(\rho, \boldsymbol{v}))$. When $\boldsymbol{v}$ is fixed, we use $\underline{f}^{(\boldsymbol{v})}(u) \equiv \underline{f}(u, \boldsymbol{v})$.

From Lemma 1, we obtain that $\forall \boldsymbol{v},\ \underline{f}^{(\boldsymbol{v})}(u)$ is *left-continuous*, that is:

Corollary 1. $\forall \boldsymbol{v} \forall \rho \in (0,1],\ \lim_{\eta \to \rho^-} \underline{f}^{(\boldsymbol{v})}(\eta) = \underline{f}^{(\boldsymbol{v})}(\rho)$.

Lemma 2. *A nonnegative unbiased estimator $\hat{f}$ must satisfy*

$$\forall \boldsymbol{v}, \forall \rho, \int_\rho^1 \hat{f}(u, \boldsymbol{v}) du \le \underline{f}^{(\boldsymbol{v})}(\rho) \tag{3}$$

Proof. Unbiased and nonnegative $\hat{f}$ must satisfy

$$\forall \boldsymbol{v}, \forall \rho \in (0,1].\ \int_\rho^1 \hat{f}(u, \boldsymbol{v}) du \le \int_0^1 \hat{f}(u, \boldsymbol{v}) du = f(\boldsymbol{v})\ . \tag{4}$$

From definition of $\underline{f}$, for all $\epsilon > 0$ and ρ, there is a vector $\boldsymbol{z}^{(\epsilon)} \in S(\rho, \boldsymbol{v})$ such that $f(\boldsymbol{z}^{(\epsilon)}) \le \underline{f}(\rho, \boldsymbol{v}) + \epsilon$. Recall that for all $u \ge \rho$, $S(u, \boldsymbol{v}) = S(u, \boldsymbol{z}^{(\epsilon)})$, hence, using, (4),

$$\int_\rho^1 \hat{f}(u, \boldsymbol{v}) du = \int_\rho^1 \hat{f}(u, \boldsymbol{z}^{(\epsilon)}) du \le f(\boldsymbol{z}^{(\epsilon)}) \le \underline{f}(\rho, \boldsymbol{v}) + \epsilon\ .$$

Taking the limit as $\epsilon \to 0$ we obtain $\int_\rho^1 \hat{f}(u, \boldsymbol{v}) du \le \underline{f}(\rho, \boldsymbol{v})$. □

The lower hull of the lower bound function and $\boldsymbol{v}$-optimality: We denote the function corresponding to the lower boundary of the convex hull (lower hull) of $\underline{f}^{(\boldsymbol{v})}$ by $H_f^{(\boldsymbol{v})}$. Our interest in the lower hull is due to the following relation The proof is postponed to the full version `http://arxiv.org/abs/1206.5637`:

Theorem 1. *An estimator $\hat{f}$ is $\boldsymbol{v}$-optimal if and only if for $u \in [0,1]$ almost everywhere*

$$\hat{f}(u, \boldsymbol{v}) = -\frac{dH_f^{(\boldsymbol{v})}(u)}{du}\ .$$

Moreover, when an unbiased and nonnegative estimator exists for f, there also exists, for any data $\boldsymbol{v}$, a nonnegative and unbiased $\boldsymbol{v}$-optimal estimator.

We use the notation $\hat{f}^{(\boldsymbol{v})}(u) = -\frac{dH_f^{(\boldsymbol{v})}(u)}{du}$ for the *$\boldsymbol{v}$-optimal estimates on outcomes consistent with $\boldsymbol{v}$*. Since the lower bound function is monotone non-increasing, so is $H_f^{(\boldsymbol{v})}$, and therefore $H_f^{(\boldsymbol{v})}$ is differentiable almost everywhere and $\hat{f}^{(\boldsymbol{v})}$ is defined almost everywhere. Figure 3 illustrates an example lower bound function and the corresponding lower hull.

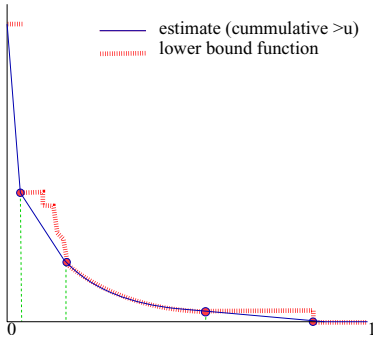


Fig. 3. Lower bound function $\underline{f}^{(\boldsymbol{v})}(u)$ for $u \in (0, 1]$ and corresponding lower hull $H_f^{(\boldsymbol{v})}(u)$, which is also the integral of the nonnegative estimator with minimum variance on $\boldsymbol{v}$: $\int_u^1 \hat{f}^{(\boldsymbol{v})}(x)dx$. The figure visualizes the lower bound function which is always left-continuous and monotone non increasing. The lower hull is continuous and also monotone non-increasing.

5 Characterization

Theorem 2. *$f \geq 0$ has an estimator that is*

- *unbiased and nonnegative* $\iff$

$$\forall \boldsymbol{v} \in \mathbf{V},\ \lim_{u \to 0^+} \underline{f}^{(\boldsymbol{v})}(u) = f(\boldsymbol{v})\ . \tag{5}$$

- *unbiased, nonnegative, and finite variances* $\iff$

$$\forall \boldsymbol{v} \in \mathbf{V},\ \int_0^1 \left(\frac{dH_f^{(\boldsymbol{v})}(u)}{du}\right)^2 du < \infty\ . \tag{6}$$

- *unbiased, nonnegative, and bounded* $\iff$

$$\forall \boldsymbol{v} \in \mathbf{V},\ \lim_{u \to 0^+} \frac{f(\boldsymbol{v}) - \underline{f}^{(\boldsymbol{v})}(u)}{u} < \infty\ . \tag{7}$$

We establish sufficiency in Theorem 2 by constructing an estimator $\hat{f}^{(J)}$ (the J estimator) that is unbiased and nonnegative when (5) holds, bounded when (7) holds, and has finite variances if (6) holds. The proof of the theorem is provided in Section 7, following the presentation of the J estimator in the next section.

6 The J Estimator

Fixing $\boldsymbol{v}$, we define an estimator $\hat{f}^{(J)}(u, \boldsymbol{v})$ incrementally, starting with $u = 1$ and such that the value at u depends on values at $u' > u$. We first define $\hat{f}^{(J)}(u, \boldsymbol{v})$ for all $u \in (\frac{1}{2}, 1]$ by $\hat{f}^{(J)}(u, \boldsymbol{v}) = 2\underline{f}(1, \boldsymbol{v})$. At each step we consider intervals of the form $(2^{-j-1}, 2^{-j}]$, setting the estimate to the same value for all outcomes $S(u, \boldsymbol{v})$ for $u \in (2^{-j-1}, 2^{-j}]$. Assuming the estimator is defined for $u \geq 2^{-j}$, we extend the definition to the interval $u \in (2^{-j-1}, 2^{-j}]$ as follows.

$$\hat{f}^{(J)}(u, \boldsymbol{v}) = 0 \text{ , } \textbf{if } \underline{f}(2^{-j}, \boldsymbol{v}) = \int_{2^{-j}}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du$$

$$\hat{f}^{(J)}(u, \boldsymbol{v}) = 2^{j+1}\left(\underline{f}(2^{-j}, \boldsymbol{v}) - \int_{2^{-j}}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du\right) \text{ , } \textbf{otherwise}$$

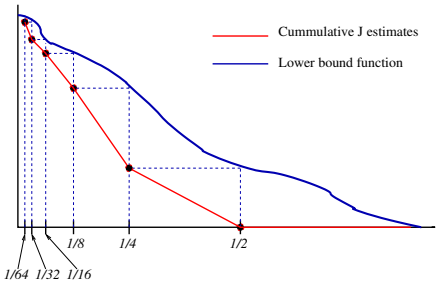


Fig. 4. Lower bound function $\underline{f}^{(\boldsymbol{v})}(u)$ for $u \in (0, 1]$ and cummulative J estimates on outcomes consistent with $\boldsymbol{v}$ $\int_u^1 \hat{f}^{(J)}(x, \boldsymbol{v}) dx$. The J estimate $\hat{f}^{(J)}(u, \boldsymbol{v})$ is the negated slope.

Lemma 3. *The J estimator is well defined, is unbiased and nonnegative when* (5) *holds, and satisfies*

$$\forall \rho \forall \boldsymbol{v}, \int_{\rho}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du \leq \underline{f}(\rho, \boldsymbol{v}) \tag{8}$$

$$\forall \rho \forall \boldsymbol{v}, \int_{\rho}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du \geq \underline{f}(4\rho, \boldsymbol{v}) \text{ .} \tag{9}$$

Proof. We first argue that the constructions, which are presented relative to a particular choices of the data $\boldsymbol{v}$, produce a consistent estimator. For that, we have to show that for every outcome $S(\rho, \boldsymbol{v})$, the assigned value is the same for all vectors $\boldsymbol{z} \in V^*(\rho, \boldsymbol{v})$. Since $\underline{f}(\rho, \boldsymbol{v}) = \underline{f}(\rho, \boldsymbol{z})$ for all $\boldsymbol{z} \in V^*(\rho, \boldsymbol{v})$, in particular this holds for $\rho = 2^{-j}$, so the setting of the estimator for $u \in (2^{-j-1}, 2^{-j}]$ is the same for all $S(u, \boldsymbol{z})$ where $\boldsymbol{z} \in V^*(2^{-j}, \boldsymbol{v})$ (also when $\boldsymbol{z} \notin V^*(2^{-j-1}, \boldsymbol{v})$). Therefore, the resulting estimator is consistently defined.

We show that the construction maintains the following invariant for $j \geq 1$:

$$\underline{f}(2^{-j+1}, \boldsymbol{v}) = \int_{2^{-j}}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du \; . \tag{10}$$

From the first step of the construction,

$$\int_{1/2}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du = \int_{1/2}^{1} 2\underline{f}(1, \boldsymbol{v}) du = \underline{f}(1, \boldsymbol{v}) \; .$$

So (10) holds for $j = 1$. Now we assume by induction that (10) holds for j and establish that it holds for $j + 1$. If $\underline{f}(2^{-j}, \boldsymbol{v}) = \underline{f}(2^{-j+1}, \boldsymbol{v})$, then by the definition of the J estimator $\int_{2^{-j-1}}^{2^{-j}} \hat{f}^{(J)}(u, \boldsymbol{v}) du = 0$ and we get

$$\int_{2^{-j-1}}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du = \int_{2^{-j}}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du = \underline{f}(2^{-j+1}, \boldsymbol{v}) = \underline{f}(2^{-j}, \boldsymbol{v}) \; .$$

Otherwise, by definition, $\int_{2^{-j-1}}^{2^{-j}} \hat{f}^{(J)}(u, \boldsymbol{v}) du = \underline{f}(2^{-j}, \boldsymbol{v}) - \underline{f}(2^{-j+1}, \boldsymbol{v})$ and hence $\int_{2^{-j-1}}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du = \underline{f}(2^{-j}, \boldsymbol{v})$ and (10) holds for $j + 1$.

From monotonicity, $\underline{f}(2^{-j+1}, \boldsymbol{v}) \leq \underline{f}(2^{-j}, \boldsymbol{v})$ and when substituting (10) in the definition of the estimator we obtain that the estimates are always nonnegative.

To establish (8) we use (10), the relation $2^{\lfloor \log_2 \rho \rfloor} \leq \rho < 2^{1+\lfloor \log_2 \rho \rfloor}$, and monotonicity of $\underline{f}(u, \boldsymbol{v})$, to obtain

$$\int_{\rho}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du \leq \int_{2^{\lfloor \log_2 \rho \rfloor}}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du = \underline{f}(2^{1+\lfloor \log_2 \rho \rfloor}, \boldsymbol{v}) \leq \underline{f}(\rho, \boldsymbol{v}) \; .$$

Similarly, we establish (9) using (10), and the relation $2^{-1+\lceil \log_2 \rho \rceil} \leq \rho \leq 2^{\lceil \log_2 \rho \rceil}$:

$$\int_{\rho}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du \geq \int_{2^{\lceil \log_2 \rho \rceil}}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du = \underline{f}(2^{1+\lceil \log_2 \rho \rceil}, \boldsymbol{v}) \geq \underline{f}(4\rho, \boldsymbol{v})$$

Lastly, unbiasedness follows from (5) and combining (8) and (9):

$$\underline{f}(\rho, \boldsymbol{v}) \geq \int_{\rho}^{1} \hat{f}^{(J)}(u, \boldsymbol{v}) du \geq \underline{f}(4\rho, \boldsymbol{v}) \; .$$

when we take the limit as $\rho \to 0$. □

Computing the J estimate from an outcome S: From the outcome we know the seed value ρ and the lower bound function $\underline{f}^{(\boldsymbol{v})}(u)$ for all $u \geq \rho$ (recall that the lower bound on this range is the same for all data $\boldsymbol{v} \in V^*(S)$, so we do not need to know the data $\boldsymbol{v}$). We compute $i \leftarrow \lfloor -\log_2 \rho \rfloor$ and use the invariant (10) in the definition of J, obtaining the J estimate $2^{i+1}(\underline{f}(2^{-i}, \boldsymbol{v}) - \underline{f}(2^{-i+1}, \boldsymbol{v}))$.

Example: We demonstrate the application of the J estimator through a simple example. The data domain in our example includes pairs (v_1, v_2) of nonnegative reals. We are interested in $f(v_1, v_2) = (\max\{v_1 - v_2, 0\})^2$, which sum aggregates to (the square of) the "one sided" Euclidean distance. The data is PPS sampled with threshold $\tau = 1$ for both entries, therefore, the sampling probability of entry i is $\min\{1, v_i\}$. Sampling is coordinated, which means that for a seed $\rho \in U[0,1]$, entry i is sampled if and only if $v_i \geq \rho$. The outcome S includes the values of the sampled entries and the seed value ρ. If no entry is sampled, or only the second entry is sampled, the lower bound function for $x \geq \rho$ is 0 and the J estimate is 0. If only the first entry is sampled, the lower bound function, for $x \geq \rho$, and accordingly, the J estimate are

$$\underline{f}^{(\boldsymbol{v})}(x) = \max\{0, v_1 - x\}^2$$
$$\hat{f}^{(J)}(S) = 2^{\lfloor -\log_2 \rho \rfloor + 1}\left(\max\{0, v_1 - 2^{-\lfloor -\log_2 \rho \rfloor}\}^2 - \max\{0, v_1 - 2^{1-\lfloor -\log_2 \rho \rfloor}\}^2 \right).$$

If both entries are sampled, the lower bound function for $x \geq \rho$ and the J estimate are

$$\underline{f}^{(\boldsymbol{v})}(x) = \max\{0, v_1 - \max\{v_2, x\}\}^2$$
$$\hat{f}^{(J)}(S) = 2^{\lfloor -\log_2 \rho \rfloor + 1}\left(\max\{0, v_1 - \max\{v_2, 2^{-\lfloor -\log_2 \rho \rfloor}\}\}^2 - \max\{0, v_1 - \max\{v_2, 2^{1-\lfloor -\log_2 \rho \rfloor}\}\}^2 \right).$$

6.1 Competitiveness of the J Estimator

Theorem 3. *The estimator $\hat{f}^{(J)}$ is $O(1)$-competitive.*

Proof. We will show that

$$\forall \boldsymbol{v}, \int_0^1 \left(\hat{f}^{(J)}(u, \boldsymbol{v})\right)^2 du \leq 84 \int_0^1 \left(\hat{f}^{(\boldsymbol{v})}(u)\right)^2 du.$$

Let $\rho = 2^{-j}$ for some integer $j \geq 0$. Recall the construction of $\hat{f}^{(J)}$ on an interval $(\rho/2, \rho]$. The value is fixed in the interval and is either 0, if $\int_\rho^1 \hat{f}^{(J)}(u, \boldsymbol{v}) du = \underline{f}^{(\boldsymbol{v})}(\rho)$ or is $2\frac{\underline{f}^{(\boldsymbol{v})}(\rho) - \int_\rho^1 \hat{f}^{(J)}(u,\boldsymbol{v}) du}{\rho}$. Using (9) in Lemma 3, we obtain that for $u \in (\rho/2, \rho]$,

$$\hat{f}^{(J)}(u, \boldsymbol{v}) \leq 2\frac{\underline{f}^{(\boldsymbol{v})}(\rho) - \int_\rho^1 \hat{f}^{(J)}(u, \boldsymbol{v}) du}{\rho} \leq 2\frac{\underline{f}^{(\boldsymbol{v})}(\rho) - \underline{f}^{(\boldsymbol{v})}(4\rho)}{\rho}$$

Thus,

$$\int_{\rho/2}^{\rho} \left(\hat{f}^{(J)}(u, \boldsymbol{v})\right)^2 du \leq \frac{\rho}{2}\frac{4}{\rho^2}\left(\underline{f}^{(\boldsymbol{v})}(\rho) - \underline{f}^{(\boldsymbol{v})}(4\rho)\right)^2 = \frac{2}{\rho}\left(\underline{f}^{(\boldsymbol{v})}(\rho) - \underline{f}^{(\boldsymbol{v})}(4\rho)\right)^2 \tag{11}$$

We now bound the expectation of $\hat{f}^{(\boldsymbol{v})}$ on $u \in (\rho/2, 4\rho)$ from below.

$$\int_{\rho/2}^{4\rho} \hat{f}^{(\boldsymbol{v})}(u)du = \int_{\rho}^{4\rho} \hat{f}^{(\boldsymbol{v})}(u)du + \int_{\rho/2}^{\rho} \hat{f}^{(\boldsymbol{v})}(u)du$$

$$\geq \int_{\rho}^{4\rho} \hat{f}^{(\boldsymbol{v})}(u)du + \frac{\rho}{2}\hat{f}^{(\boldsymbol{v})}(\rho) \tag{12}$$

$$\geq \int_{\rho}^{4\rho} \hat{f}^{(\boldsymbol{v})}(u)du + \frac{\rho}{2}\frac{\underline{f}^{(\boldsymbol{v})}(\rho) - \int_{\rho}^{1} \hat{f}^{(\boldsymbol{v})}(u)du}{\rho} \tag{13}$$

$$= \int_{\rho}^{4\rho} \hat{f}^{(\boldsymbol{v})}(u)du + + \frac{1}{2}\left(\underline{f}^{(\boldsymbol{v})}(\rho) - \int_{\rho}^{4\rho} \hat{f}^{(\boldsymbol{v})}(u)du - \int_{4\rho}^{1} \hat{f}^{(\boldsymbol{v})}(u)du\right)$$

$$= \frac{1}{2}\int_{\rho}^{4\rho} \hat{f}^{(\boldsymbol{v})}(u)du + \frac{1}{2}\left(\underline{f}^{(\boldsymbol{v})}(\rho) - \int_{4\rho}^{1} \hat{f}^{(\boldsymbol{v})}(u)du\right) \geq \frac{1}{2}\left(\underline{f}^{(\boldsymbol{v})}(\rho) - \underline{f}^{(\boldsymbol{v})}(4\rho)\right) \tag{14}$$

Inequality (12) follows from monotonicity of the function $\hat{f}^{(\boldsymbol{v})}$. Inequality (13) from the definition of $\hat{f}^{(\boldsymbol{v})}$ as the negated derivative of the lower hull of $\underline{f}^{(\boldsymbol{v})}$ $\hat{f}^{(\boldsymbol{v})}(\rho) \geq \frac{\underline{f}^{(\boldsymbol{v})}(\rho) - \int_{\rho}^{1} \hat{f}^{(\boldsymbol{v})}(u)du}{\rho}$ (More precisely, we can use the explicit definition of the $\boldsymbol{v}$-optimal estimates provided in the full version, $\hat{f}^{(\boldsymbol{v})}(\rho) = \inf_{0\leq\eta<\rho} \frac{\underline{f}^{(\boldsymbol{v})}(\eta) - \int_{\rho}^{1} \hat{f}(u,\boldsymbol{v})du}{\rho-\eta} \geq \inf_{0\leq\eta<\rho} \frac{\underline{f}^{(\boldsymbol{v})}(\rho) - \int_{\rho}^{1} \hat{f}(u,\boldsymbol{v})du}{\rho-\eta} = \frac{\underline{f}^{(\boldsymbol{v})}(\rho) - \int_{\rho}^{1} \hat{f}(u,\boldsymbol{v})du}{\rho}$). Lastly, inequality (14) uses $\int_{4\rho}^{1} \hat{f}^{(\boldsymbol{v})}(u)du \leq \underline{f}^{(\boldsymbol{v})}(4\rho)$, which follows from nonnegativity of $\hat{f}^{(\boldsymbol{v})}$ and Lemma 2.

Dividing both side by 3.5ρ we obtain a lower bound on the *average* value of $\hat{f}^{(\boldsymbol{v})}(u)$ in the interval $[\rho/2, 4\rho]$.

$$\frac{1}{3.5\rho}\int_{\rho/2}^{4\rho} \hat{f}^{(\boldsymbol{v})}(u)du \geq \frac{1}{7\rho}\left(\underline{f}^{(\boldsymbol{v})}(\rho, \boldsymbol{v}) - \underline{f}^{(\boldsymbol{v})}(4\rho)\right) \tag{15}$$

We next show that the value $\hat{f}^{(J)}(u, \boldsymbol{v})$ on $u \in (\rho/2, \rho]$ is at most some constant times the expected value of the square of $\hat{f}^{(\boldsymbol{v})}$ on $u \in (\rho/2, 4\rho)$.

$$\int_{\rho/2}^{4\rho} \hat{f}^{(\boldsymbol{v})}(u)^2 du \geq \int_{\rho/2}^{4\rho} \left(\frac{1}{3.5\rho}\int_{\rho/2}^{4\rho} \hat{f}^{(\boldsymbol{v})}(u)du\right)^2 du \geq \tag{16}$$

$$\int_{\rho/2}^{4\rho} \left(\frac{1}{7\rho}\left(\underline{f}^{(\boldsymbol{v})}(\rho) - \underline{f}^{(\boldsymbol{v})}(4\rho)\right)\right)^2 du \geq \tag{17}$$

$$3.5\rho\left(\frac{1}{7\rho}\right)^2\left(\underline{f}^{(\boldsymbol{v})}(\rho) - \underline{f}^{(\boldsymbol{v})}(4\rho)\right)^2 \geq \tag{18}$$

$$\frac{1}{28}\int_{\rho/2}^{\rho} \hat{f}^{(J)}(u, \boldsymbol{v})^2 du$$

Inequality (16) uses the fact that for any random variable X, $(\mathsf{E}[X])^2 \leq \mathsf{E}[X^2]$ applied to $\hat{f}^{(\boldsymbol{v})}$ for $u \in (\rho/2, 4\rho]$. Inequality (17) follows from (15). Lastly, Inequality (18) follows from (11). We obtain

$$\int_0^1 \hat{f}^{(J)}(u)^2 du = \sum_{i=0}^{\infty} \int_{2^{-i-1}}^{2^{-i}} \hat{f}^{(J)}(u)^2 du$$
$$\leq 28 \sum_{i=0}^{\infty} \int_{2^{-i-1}}^{\min\{1,2^{-i+2}\}} \hat{f}^{(\boldsymbol{v})}(u)^2 du \leq 28 \cdot 3 \int_0^1 \hat{f}^{(\boldsymbol{v})}(u)^2 du$$

□

7 Proof of Theorem 2

Proof. • **"⇒" (5):** From Lemma 2, an unbiased and nonnegative estimator $\hat{f}$ must satisfy (3). Fixing $\boldsymbol{v}$ in (3) and taking the limit as $\rho \to 0$ we obtain that $\mathsf{E}[\hat{f}|\boldsymbol{v}] = \int_0^1 \hat{f}(u,\boldsymbol{v})du \leq \lim_{u\to 0} \underline{f}^{(\boldsymbol{v})}(u,\boldsymbol{v})$. Combining with unbiasedness: $\mathsf{E}[\hat{f}|\boldsymbol{v}] = f(\boldsymbol{v})$ we obtain (5).

• **"⇐" (5):** Follows immediately from Lemma 3.

• **"⇒" (7):** We bound from below the contribution to the expectation of unbiased and nonnegative $\hat{f}$ of outcomes $S(u,\boldsymbol{v})$ for $u \leq \rho$: $\int_0^\rho \hat{f}(u,\boldsymbol{v}) = \int_0^1 \hat{f}(u,\boldsymbol{v}) - \int_\rho^1 \hat{f}(u,\boldsymbol{v}) \geq f(\boldsymbol{v}) - \underline{f}^{(\boldsymbol{v})}(u)$. The last inequality follows from unbiasedness and nonnegativity (3). Hence, the average value $\hat{f}(u,\boldsymbol{v})$ when $u < \rho$ must be at least $\frac{f(\boldsymbol{v})-\underline{f}^{(\boldsymbol{v})}(\rho)}{\rho}$, and thus, considering all possible values of $\rho > 0$, we obtain that $\hat{f}$ can be bounded only if it satisfies (7).

• **"⇐" (7):** Note that (7) $\implies$ (5), and therefore the conditions of Lemma 3 are satisfied and the J estimator is well defined, nonnegative, and unbiased. It remains to show that given (7), or the equivalent statement

$$\forall \boldsymbol{v}\ \exists c < \infty\ \forall u,\ f(v) - \underline{f}^{(\boldsymbol{v})}(u) \leq cu\ , \tag{19}$$

the J estimator is bounded. Fix $\boldsymbol{v}$ and let c be as in (19).

$$\hat{f}^{(J)}(\rho,\boldsymbol{v}) \leq 2\frac{\underline{f}^{(\boldsymbol{v})}(\rho/2) - \int_{2\rho}^1 \hat{f}^{(J)}(u,\boldsymbol{v})du}{\rho} \tag{20}$$
$$\leq 2\frac{f(\boldsymbol{v}) - \underline{f}^{(\boldsymbol{v})}(8\rho)}{\rho} \tag{21}$$
$$= 16\frac{f(\boldsymbol{v}) - \underline{f}^{(\boldsymbol{v})}(8\rho)}{8\rho} \leq 16c \tag{22}$$

Inequality (20) is from the definition of the J estimator. Inequality (21) uses definition of the lower bound function and (9). Lastly, (22) follows from our assumption (19).

• **" $\iff$ "** (6): From Theorem 1, for all $\boldsymbol{v}$, (6), which is square-integrability of $\hat{f}^{(\boldsymbol{v})}(u)$, is *necessary* for existence of a nonnegative unbiased estimator with finite variance for $\boldsymbol{v}$. Sufficiency follows from the proof of Theorem 3, which shows that for all $\boldsymbol{v}$, the expectation of the square of the J estimator is at most a constant times the minimum possible, and (2), which states that the variance is bounded if and only if the expectation of the square is bounded. □

Conclusion

We developed a precise understanding of the queries we can estimate accurately over coordinated samples, defined variance competitiveness, and showed that it is generally attainable. Our work uses a fresh, CS-inspired, and unified approach to the study of estimators that is particularly suitable for data analysis from samples.

In a follow up work [12], we study the interaction of competitiveness and *variance optimality* (an estimator is variance optimal of it can not be strictly improved), tighter competitiveness bounds, and propose natural estimators. We also plan to extend our initial treatment of independent sampling [11].

On the applied front, our work is motivated by the prevalent use of sampling as synopsis of large data sets. We demonstrate its potential for difference queries in a companion experimental paper [13]. We also plan to apply it for analysis of massive graphs. In the long run, we envision automated tools that provide estimates according to specifications of the sampling scheme, query, competitive ratio, and prioritization of patterns in the data.

References

1. Beyer, K.S., Haas, P.J., Reinwald, B., Sismanis, Y., Gemulla, R.: On synopses for distinct-value estimation under multiset operations. In: SIGMOD, pp. 199–210. ACM (2007)
2. Brewer, K.R.W., Early, L.J., Joyce, S.F.: Selecting several samples from a single population. Australian Journal of Statistics 14(3), 231–239 (1972)
3. Broder, A.Z.: On the resemblance and containment of documents. In: Proceedings of the Compression and Complexity of Sequences, pp. 21–29. IEEE (1997)
4. Broder, A.: Identifying and filtering near-duplicate documents. In: Giancarlo, R., Sankoff, D. (eds.) CPM 2000. LNCS, vol. 1848, pp. 1–10. Springer, Heidelberg (2000)
5. Byers, J.W., Considine, J., Mitzenmacher, M., Rost, S.: Informed content delivery across adaptive overlay networks. IEEE/ACM Trans. Netw. 12(5), 767–780 (2004)
6. Cohen, E.: Size-estimation framework with applications to transitive closure and reachability. J. Comput. System Sci. 55, 441–453 (1997)
7. Cohen, E., Kaplan, H.: Spatially-decaying aggregation over a network: model and algorithms. J. Comput. System Sci. 73, 265–288 (2007)
8. Cohen, E., Kaplan, H.: Summarizing data using bottom-k sketches. In: Proc. of ACM PODC (2007)
9. Cohen, E., Kaplan, H.: Tighter estimation using bottom-k sketches. In: VLDB (2008)
10. Cohen, E., Kaplan, H.: Leveraging discarded samples for tighter estimation of multiple-set aggregates. In: ACM SIGMETRICS (2009)
11. Cohen, E., Kaplan, H.: Get the most out of your sample: Optimal unbiased estimators using partial information. In: Proc. of ACM PODS (2011), full version: `http://arxiv.org/abs/1203.4903`
12. Cohen, E., Kaplan, H.: A case for customizing estimators: Coordinated samples. Technical Report cs.ST/1212.0243, arXiv (2012)
13. Cohen, E., Kaplan, H.: How to estimate change from samples. Technical Report cs.DS/1203.4903, arXiv (2012)
14. Cohen, E., Kaplan, H., Sen, S.: Coordinated weighted sampling for estimating aggregates over multiple weight assignments. In: VLDB (2009), full version: `http://arxiv.org/abs/0906.4560`

15. Cohen, E., Wang, Y.-M., Suri, G.: When piecewise determinism is almost true. In: Proc. Pacific Rim International Symposium on Fault-Tolerant Systems (1995)
16. Das, A., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: WWW (2007)
17. Duffield, N., Thorup, M., Lund, C.: Priority sampling for estimating arbitrary subset sums. J. Assoc. Comput. Mach. 54(6) (2007)
18. Efraimidis, P.S., Spirakis, P.G.: Weighted random sampling with a reservoir. Inf. Process. Lett. 97(5), 181–185 (2006)
19. Gibbons, P., Tirthapura, S.: Estimating simple functions on the union of data streams. In: ACM SPAA (2001)
20. Gibbons, P.B.: Distinct sampling for highly-accurate answers to distinct values queries and event reports. In: VLDB (2001)
21. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB (1999)
22. Hadjieleftheriou, M., Yu, X., Koudas, N., Srivastava, D.: Hashed samples: Selectivity estimators for set similarity selection queries. In: VLDB (2008)
23. Hájek, J.: Sampling from a finite population. Marcel Dekker, New York (1981)
24. Horvitz, D.G., Thompson, D.J.: A generalization of sampling without replacement from a finite universe. Journal of the American Statistical Association 47(260), 663–685 (1952)
25. Indyk, P.: Stable distributions, pseudorandom generators, embeddings and data stream computation. In: IEEE FOCS (2001)
26. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: ACM STOC (1998)
27. Lanke, J.: On umv-estimators in survey sampling. Metrika 20(1), 196–202 (1973)
28. Mosk-Aoyama, D., Shah, D.: Computing separable functions via gossip. In: ACM PODC (2006)
29. Ohlsson, E.: Sequential poisson sampling. J. Official Statistics 14(2), 149–162 (1998)
30. Ohlsson, E.: Coordination of pps samples over time. In: The 2nd International Conference on Establishment Surveys, pp. 255–264. American Statistical Association (2000)
31. Rosén, B.: Asymptotic theory for successive sampling with varying probabilities without replacement, I. The Annals of Mathematical Statistics 43(2), 373–397 (1972)
32. Rosén, B.: Asymptotic theory for order sampling. J. Statistical Planning and Inference 62(2), 135–158 (1997)
33. Saavedra, P.J.: Fixed sample size pps approximations with a permanent random number. In: Proc. of the Section on Survey Research Methods, Alexandria, VA, pp. 697–700. American Statistical Association (1995)
34. Szegedy, M.: The DLT priority sampling is essentially optimal. In: ACM STOC (2006)

Robust Randomness Amplifiers: Upper and Lower Bounds

Matthew Coudron[1], Thomas Vidick[1,2], and Henry Yuen[1]

[1] MIT CSAIL*
[2] Centre for Quantum Technologies, Singapore

Abstract. A recent sequence of works, initially motivated by the study of the nonlocal properties of entanglement, demonstrate that a source of information-theoretically certified randomness can be constructed based only on two simple assumptions: the prior existence of a short random seed and the ability to ensure that two black-box devices do not communicate (i.e. are non-signaling). We call protocols achieving such certified amplification of a short random seed *randomness amplifiers*. We introduce a simple framework in which we initiate the systematic study of the possibilities and limitations of randomness amplifiers. Our main results include a new, improved analysis of a robust randomness amplifier with exponential expansion, as well as the first upper bounds on the maximum expansion achievable by a broad class of randomness amplifiers. In particular, we show that non-adaptive randomness amplifiers that are robust to noise cannot achieve more than doubly exponential expansion. Finally, we show that a wide class of protocols based on the use of the CHSH game can only lead to (singly) exponential expansion if adversarial devices are allowed the full power of non-signaling strategies. Our upper bound results apply to all known non-adaptive randomness amplifier constructions to date.

1 Introduction

Consider the following simple game, called the CHSH game: a referee sends each of a pair of isolated, cooperating but non-communicating players Alice and Bob a bit $x, y \in \{0, 1\}$ respectively, chosen uniformly at random. Alice and Bob reply with bits $a, b \in \{0, 1\}$, and they win the game iff $a \oplus b = x \wedge y$. If Alice and Bob employ classical strategies, the probability that they win the game is at most 75%. As a consequence, one readily sees that *any* non-signaling strategy (i.e. a strategy in which each player's marginal output distribution is independent of the other player's input) that wins the CHSH game with probability strictly

* Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology. T.V. was supported by the National Science Foundation under Grant No. 0844626 and by and by the Ministry of Education, Singapore under the Tier 3 grant MOE2012-T3-1-009. H.Y. was supported by an NSF Graduate Fellowship Grant No. 1122374 and National Science Foundation Grant No. 1218547. M.C. was supported by the National Science Foundation under Grant No. 0801525.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 468–483, 2013.

larger than 75% *must* generate randomness. Remarkably, there actually *exists* such a strategy, allowing them to win with probability $\cos^2(\pi/8) \approx 85\%$. Furthermore, the strategy can be physically implemented using simple "everyday" quantum mechanical devices that utilize shared entanglement [2]. In his Ph.D. thesis, Colbeck [6] was the first to explicitly observe that the CHSH game could be interpreted as a simple *statistical test* for the presence of randomness: the test repeatedly "plays" the CHSH game with a given pair of black-box devices. Provided that non-signaling is enforced between the devices (via space-time separation or other means), the observation of a sufficiently high success probability can be used to *certify* the generation of "fresh" randomness. In particular, the soundness of the test does *not* require one to assume that quantum mechanics is correct. (Of course, as far as we know, the easiest way to actually *pass* the test is to perform certain specific quantum mechanical measurements on two halves of an EPR pair!)

It is easy to see that without any assumptions, black-box randomness testing is impossible: if a (randomized) test T accepts a random source X with some probability p, by linearity of expectation there automatically exists a deterministic source Y (i.e. a fixed string) that is accepted with probability at least p. Thus it is quite surprising that a very simple physical assumption – that it is possible to enforce non-signaling between two devices – allows for an information-theoretic method to test for randomness in the devices' outputs. As we shall see, the test provides guarantees on the *min-entropy* of the outputs, which enables the tester to later apply a classical procedure such as a randomness extractor to generate bits that are nearly independent and uniformly distributed, making them useful in algorithmic or cryptographic applications (for a survey on randomness extractors we refer to [18]).

Starting with work of Pironio et al. [14], a series of papers [7,11,19,15] have demonstrated that not only can randomness be certified, but it can be *expanded* as well. In [14], a protocol was given in which the testing requires m bits of seed randomness, but the output of the devices is certified to have $\Omega(m^2)$ bits of min-entropy. Vazirani and Vidick [19] show that there exists a protocol that can produce $2^{\Omega(m)}$ bits of certifiable randomness starting from m bits of seed randomness. In their protocol, the referee uses the seed to generate pseudorandom inputs for the two devices; the devices play $2^{O(m \log^2 m)}$ iterations of (a variant of) the CHSH game on those inputs. The referee then tests that the wins and losses of the devices obey a simple statistical condition. Whenever the devices are designed in a way that they pass the test with non-negligible probability, their output distribution (conditioned on passing) must have high min-entropy. The test, however, is not *robust* in the sense that even a very slight deviation by the devices from the intended behavior will result in rejection. Robust protocols for exponential randomness expansion were devised in [11,15] but they use *two* pairs of devices, and furthermore rely on the strong assumption that there is no entanglement between the pairs.

These prior works immediately raise a wealth of questions, for which there has been no systematic investigation so far: What is the maximal expansion

achievable? Could doubly exponential expansion, or even an *unbounded*, expansion of randomness be possible? Can exponential expansion be achieved using a more natural protocol that is robust to noise? What are the minimal assumptions required on the seed quality? While many specific protocols have been considered in the quantum information literature [7,11,8], to our knowledge no general model of randomness certification and amplification had yet been formulated.

In this paper we introduce a simple and natural framework for randomness amplification which captures nearly all previously considered protocols. We initiate the systematic investigation of the possibilities and limitations of such protocols, which we call *randomness amplifiers*.[1] In particular, we present both the first *upper bounds* on the achievable randomness expansion of natural protocols, as well as the first robust exponential *lower bounds*. (Note that here, contrary to common usage in theoretical computer science, upper bounds on randomness expansion are *impossibility* results, whereas lower bounds are *possibility* results.)

Robust Protocols. An appealing aspect of randomness amplifiers is that they only rely on two basic physical assumptions: the ability to enforce the non-signaling condition between devices, and the a priori existence of a some small amount of randomness to use as seed. As such, these protocols lend themselves quite naturally to experimental implementations. In fact, [14] report an implementation of their quadratic randomness amplifier in which 42 bits of certified randomness were generated (over the course of a month of experiments!). However, noise as well as errors due to imperfections in laboratory equipment are unavoidable in such experiments. Given the recent interest in realizing efficient implementations of randomness expansion protocols[2], it is important to understand the power and limitations of protocols that behave robustly in the presence of noise and imperfect devices. Some randomness amplifiers, such as the one in [19], are not robust to noise. Is this an artifact or an intrinsic limitation of protocols that achieve exponential randomness expansion?

Our Results

The Model. Our first contribution is the introduction of a natural model for randomness amplifiers. Abstractly, we think of a randomness amplifier as a family of *protocols* describing an interaction between a trusted entity (called the referee) and a pair of black-box devices. The referee selects inputs to the devices, collects outputs, and based on these decides to either accept or reject the devices' outputs. The protocols are parametrized by a *seed length* m, which is the amount of initial randomness required to execute the protocol. The output of the protocol is defined as the output of the black-box devices over the course of the interaction (provided the referee accepted). The procedure has completeness

[1] These protocols have been called "randomness expanders" or "randomness expansion protocols" in prior works, but we adopt the term randomness amplifiers to avoid confusion with the traditional concept of expanders.

[2] Such protocols have recently been suggested as a benchmark for the closure of the so-called *detection loophole*. We refer to the recent survey [3] for more details.

c, soundness s, and expansion $g = g(m)$ if (i) there exists a pair of non-signaling devices, called the *ideal devices*, such that the referee's interaction with them will result in a "pass" with probability at least c, and (ii) for *any* pair of non-signaling devices (either bound by the laws of quantum mechanics or not, depending on context) such that they pass the protocol with probability at least s, the output distribution of the devices has min-entropy at least $g(m)$ — where, ideally, $g(m) \gg m$.

The interaction between the referee and the devices could a priori be arbitrary. In this paper we restrict our attention to *non-adaptive* protocols. In such protocols the referee uses his random seed to select a pair of input strings to be given to each device. He then provides the inputs one symbol at a time, collecting outputs from the devices. At the end of the interaction, the referee applies a test to the inputs and outputs he has collected. Such protocols are called non-adaptive because the inputs to the devices do not depend on the devices' outputs in previous rounds. Nearly all protocols considered in the literature are non-adaptive.

In addition, we formalize the notion of "robust" randomness amplifiers: informally, an amplifier is robust if small deviation from the behavior of the ideal devices still results in acceptance with high probability. Naturally, allowing noisy devices makes the analysis harder, e.g. to prove lower bounds on robust protocols we have to account for the fact that devices may use the freedom to deviate to cheat the protocol. However, we will also show that in certain cases, non-robust protocols can be cheated by malicious devices that exploit the possibility for noise-free operation!

A Robust Lower Bound. Our first result is a lower bound: we extend and generalize the result of [19] by devising a randomness amplifier that attains exponential expansion and is robust to noisy devices. The underlying protocol is simple and can be based on any non-local game (and not only the CHSH game as in [19]) that is *randomness generating*. Informally, randomness generating games are such that any strategy achieving a success probability strictly higher than the classical value must produce randomized answers, on a certain fixed pair of inputs (x_0, y_0) that depend only on the game, not the strategy. Many examples of games are known to be randomness generating, including the CHSH game and the so-called Magic Square game [1].

Fix a two-player game G. Let η denote the "noise tolerance" parameter, ε a target "security" parameter and R a number of rounds. The robust protocol P_G is as follows: in each round, with some small probability p_c the two devices are presented with inputs as prescribed in the game G. Such rounds are called game rounds. Otherwise, they are presented with some default inputs x_0, y_0 respectively. The referee collects the outputs of the two devices for the R rounds, and checks that on average over the game rounds the devices' inputs and outputs satisfy the game condition a fraction of times that is at least the maximum success probability achievable in G using quantum mechanics, minus η.

Theorem 1 (Informal). *Let m be a positive integer. Let G be a randomness generating game, $\eta, \varepsilon > 0$ and P_G the protocol described above, for some*

$R = R(m) \leq \exp(m/\log(1/\varepsilon))$ and $p_c = \Theta(\log(1/\varepsilon)/R)$. Then P_G uses m bits of seed, has completeness $1 - \exp(-\eta^2 R)$, soundness ε and expansion $g(m) = \Omega(R(m))$.

Upper Bounds. We present the first upper bounds on non-adaptive randomness amplifiers. Our first upper bound applies to protocols based on *perfect games*, which are games G such that there exists a quantum strategy that wins G with probability 1 (an example is the Magic Square game [1]). We consider simple protocols in which the referee's test is to verify that the devices win every single round. We give a simple argument, based on the construction of a "cheating strategy" for the devices, showing that any such protocol can achieve at most doubly exponential expansion.

While this simple class of protocols already encompasses some protocols introduced in the literature, such as one described in [6], many protocols do not use perfect games and such a stringent testing condition from the referee. We thus extend this initial upper bound and show that it also applies to arbitrary non-adaptive randomness amplifiers, provided that they are noise-robust and the ideal devices play each round independently.

Theorem 2 (Informal). *Let the family of protocols $P = (P_m)$ be a non-adaptive randomness amplifier. Suppose that for all $m \in \mathbb{N}$, P_m is noise-robust and the ideal devices for P_m play each round independently. Then, for all $m \in \mathbb{N}$ there exists two quantum devices that are accepted by the protocol P_m with high probability, but whose output min-entropy is at most $2^{O(2^m)}$.*

We refer to Theorem 5 for a precise statement. The basic idea for the cheating strategy is to show that, provided the referee's seed is short enough, the devices can often deterministically re-use some of their outputs in previous rounds. That the referee's test can be arbitrary complicates the argument somewhat, a priori preventing a systematic re-use by the devices of their past outputs: the test could check for obvious patterns that could arise in any obvious re-use strategies. To get around this we use the probabilistic method to show that for any noise-robust test there exists a randomness-efficient re-use strategy that will fool it.

Our last upper bound is a stronger, *exponential* upper bound on randomness amplifiers that are based on the CHSH game and in which the referee's test only depends on the pattern of wins and losses in the game that is observed in the protocol. However, our "cheating strategy" for such protocols requires the use of perfect non-signaling devices (which are able to win the CHSH game with probability 1). As such, the significance of the theorem is in the proof rather than in the statement: it demonstrates the possibility for elaborate cheating strategies that exploit the structure of the protocol in order to be accepted in a highly randomness-efficient way. Overcoming this kind of behavior of the devices is a major hurdle in designing protocols that achieve more than exponential expansion of randomness, a tantalizing open problem that we leave open for further work.

Related Work. As mentioned earlier, [14], building on [6], were the first to obtain a quantitative lower bound on randomness expansion. They showed that

quantum or non-signaling devices that demonstrate *any* Bell inequality violation can be used to certify randomness. Fehr et al. [11] extended this result to demonstrate exponential expansion, although their protocol requires the use of two *unentangled* pairs of devices. Vazirani and Vidick [19] describe a protocol with exponential expansion that only requires two devices. Their protocol, however, is not robust to noise and is tailored to the specifics of the CHSH game.

When considering the use of the bits generated by a randomness amplifier in a cryptographic task it may be necessary to obtain stronger guarantees than simply a lower bound on their min-entropy: indeed, in some cases it is essential that the bits not only appear random by themselves, but are also uncorrelated with any potential adversary (say, the maker of the devices). The protocol of [11] is proven secure against classical adversaries; [19] also obtain security against quantum adversaries. In this work we do not consider such extended guarantees of security.

Outline of the Paper. We start with some preliminaries in Section 2. Our model is introduced in Section 3. In Section 4 we establish our exponential lower bound, while Section 5 contains our doubly exponential and exponential upper bounds. We refer the reader to the full version of the paper [9], which contains the proofs that were omitted here due to space constraints.

2 Preliminaries

Notation. Given an integer n we write $[n] = \{1, \ldots, n\}$. Given a string $x \in \mathcal{X}^n$, where $\mathcal{X}$ is a finite alphabet, we let $x_{\leq i} = (x_1, \ldots, x_i)$, $x_{>i} = (x_{i+1}, \ldots, x_n)$, etc. If $\mathcal{X}, \mathcal{Y}$ are alphabets and π a probability distribution over $\mathcal{X} \times \mathcal{Y}$, for all $R \in \mathbb{N}$ we let $\pi^{\otimes R}$ denote the product distribution defined over $\mathcal{X}^R \times \mathcal{Y}^R$ by $\pi^{\otimes R}(x_1, \ldots, x_R, y_1, \ldots, y_R) = \prod_{i \in [R]} \pi(x_i, y_i)$. We use capital letters $X, Y, \ldots$ to denote random variables. Let X be a random variable that takes values in some discrete domain $\mathcal{D}$. Its min-entropy is defined as $H_\infty(X) = -\log \max_{x \in \mathcal{D}} \Pr(X = x)$. The Shannon entropy of a random variable X is denoted $H(X)$ as usual. We also define the max-entropy of a random variable X as $H_0(X) = \log(|\mathrm{supp}(X)|)$, where $\mathrm{supp}(X)$ denotes the support of X. The conditional min-entropy is defined as $H_\infty(X|Y) = -\log\left(\sum_y \Pr(Y = y) 2^{-H_\infty(X|Y=y)}\right)$. For two discrete random variables X, Y with the same domain, their statistical distance is $\|X - Y\|_1 = \frac{1}{2}\sum_{x \in \mathcal{D}} |\Pr(X = x) - \Pr(Y = x)|$. For $\varepsilon > 0$, the smoothed min-entropy of a discrete random variable X is defined as $H_\infty^\varepsilon(X) = \sup_{\tilde{X}, \|\tilde{X} - X\|_1 \leq \varepsilon} H_\infty(\tilde{X})$, where the supremum is taken over all $\tilde{X}$ defined on $\mathcal{D}$. The smoothed conditional min-entropy is $H_\infty^\varepsilon(X|Y) = \sup_{(\tilde{X}, \tilde{Y}), \|(\tilde{X}, \tilde{Y}) - (X, Y)\|_1 \leq \varepsilon} H_\infty(\tilde{X}|\tilde{Y})$. We also define the smooth entropy of a random variable X, conditioned on an event T, as the smooth entropy of a random variable having the distribution of X conditioned on T.

Two-Player Games. A two-player game G is specified by input alphabets $\mathcal{X}$ and $\mathcal{Y}$, output alphabets $\mathcal{A}$ and $\mathcal{B}$, an input distribution π on $\mathcal{X} \times \mathcal{Y}$, and a game

predicate $G : \mathcal{X} \times \mathcal{Y} \times \mathcal{A} \times \mathcal{B} \to \{0,1\}$. The game is played between a referee and two non-communicating players, who we typically call Alice and Bob. The referee generates inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ according to π, and sends them to Alice and Bob respectively. Alice answers with $a \in \mathcal{A}$ and Bob answers with $b \in \mathcal{B}$. The referee accepts iff $G(a,b,x,y) = 1$, in which case we say that the players win (or pass) the game.

Strategies. Given a game G, we define its *value* as the maximum winning probability of two players in the game, where the probability is taken over the referee's choice of inputs and randomness that may be part of the players' strategy. In full generality, a strategy S is specified by a family of distributions $\{p_S(\cdot,\cdot|x,y) : \mathcal{A} \times \mathcal{B} \to [0,1]\}_{(x,y)\in\mathcal{X}\times\mathcal{Y}}$, parametrized by input pairs (x,y) and defined over the output alphabet $\mathcal{A} \times \mathcal{B}$. The value of G clearly depends on restrictions that we may place on the allowed families of distributions, and we (as is customary in the study of two-player games in the quantum literature) consider three distinct restrictions:

First, if the players are restricted to classical deterministic strategies, specified by functions $f_A : \mathcal{X} \to \mathcal{A}$ for Alice and $f_B : \mathcal{Y} \to \mathcal{B}$ for Bob, we obtain the *classical value*, which is defined as $\omega_c(G) = \max_{f_A,f_B} \sum_{x,y} \pi(x,y) G(f_A(x), f_B(y), x, y)$. It is not hard to see that the use of private or even shared randomness by the players will not increase the classical value. Second, by allowing all strategies that may be implemented locally using quantum mechanics, including the use of entanglement, one obtains the *quantum value* of G, $\omega_q(G)$. In this paper we will not need to use the formalism of quantum strategies, and we refer to e.g. [5] for a good introduction. Finally, we may allow any strategy which respects the non-signaling principle: the only restriction on the players' family of distributions is that it satisfies $\forall x \in \mathcal{X}, y, y' \in \mathcal{Y}, a \in \mathcal{A}$, $p_S(a|x,y) = \sum_b p_S(a,b \mid x,y) = \sum_b p_S(a,b \mid x,y') = p_S(a|x,y')$, and a symmetric condition holds when marginalizing over the first players' output. The corresponding value is called the *non-signaling* value $\omega_{ns}(G)$. It is clear that, for any game G, $\omega_c(G) \leq \omega_q(G) \leq \omega_{ns}(G)$. Examples of games are known for which all three inequalities are strict (the CHSH game, see below). There are also games for which the first inequality is strict, and the second is an equality (e.g. the so-called Magic Square game [1]), and for which the first inequality is an equality and the second is strict (see e.g. [12]).

The CHSH Game. The CHSH game is a two-player game with two non-communicating players, Alice and Bob, who are given independent random inputs $x, y \in \{0,1\}$ respectively. Their task is to produce outputs $a, b \in \{0,1\}$ such that $a \oplus b = x \wedge y$. The classical value of CHSH is $\omega_c(\text{CHSH}) = 3/4$. There is a simple quantum strategy based on the use of a single EPR pair that demonstrates $\omega_q(\text{CHSH}) \geq \cos^2(\pi/8) \approx 85\%$, and in fact it is an optimal quantum strategy [4,13]. Furthermore, $\omega_{ns}(G) = 1$ (see [9] for a proof).

(Non-Adaptive) Protocols. Informally, a protocol prescribes the interaction between a trusted *referee* and a pair of *devices*, which we usually denote by D_A and D_B. A protocol can be thought of as a multi-round game in which the rounds are played sequentially; we use the word "devices" rather than

"players" to refer to the fact that the interaction may go on for many rounds, but there is no essential difference. In this paper, we restrict our attention to *non-adaptive* protocols, where the referee's messages to the devices are independent of the devices' outputs. Formally, a non-adaptive protocol P is specified by a tuple $\langle \mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, R, \pi, T \rangle$, where: $\mathcal{X}, \mathcal{Y}$ are finite input alphabets, $\mathcal{A}, \mathcal{B}$ are finite output alphabets, $R \in \mathbb{N}$ is the number of rounds of interaction, π is the input probability distribution over $\mathcal{X}^R \times \mathcal{Y}^R$, and $T : \mathcal{X}^R \times \mathcal{Y}^R \times \mathcal{A}^R \times \mathcal{B}^R \to \{0, 1\}$ is the referee's *test*.

Given such a protocol P, the interaction between the referee and a pair of devices (D_A, D_B) proceeds as follows: using private randomness, the referee samples the input sequence $(x, y) \in \mathcal{X}^R \times \mathcal{Y}^R$ from π. Then, for each round $i \in [R]$, the referee distributes $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ to D_A and D_B, respectively. Devices D_A and D_B are required to produce outputs $a_i \in \mathcal{A}$ and $b_i \in \mathcal{B}$, respectively. Let $a = (a_i)$ and $b = (b_i)$. After R rounds of interaction, the referee *accepts* if $T(x, y, a, b) = 1$. Otherwise, the referee *rejects*.

Given a protocol P and a pair of devices (D_A, D_B), a *strategy* for the devices is a description of their behavior in the protocol: for each round index i, a family of distributions $\{p(a_i, b_i | x_i, y_i, \mathrm{hist}_i)\}$ on $\mathcal{A}_i \times \mathcal{B}_i$, where hist_i is the *history* of the protocol prior to round i, i.e. the list of inputs and outputs generated by the devices in previous rounds. We call a strategy quantum (resp. non-signaling) if it can be implemented using isolated quantum (resp. non-signaling) devices.

3 Randomness Amplifiers

In this section we define the notion of *randomness amplifiers* that we use throughout the paper. A randomness amplifier is given by a family $(P_m)_{m \in \mathbb{N}}$ of non-adaptive protocols. The following definition summarizes the important parameters associated with a non-adaptive randomness amplifier.

Definition 1. *A family of protocols $P = (P_m) = \langle \mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, R_m, \pi_m, T_m \rangle$ is a* ***randomness amplifier*** *with* ***seed length*** *m,* ***completeness*** *$c = c(m)$,* ***soundness*** *$s = s(m) < c$ against quantum (resp. non-signaling) strategies,* ***smoothness*** *$\varepsilon = \varepsilon(m)$,* ***expansion*** *$g = g(m)$ and* ***ideal strategy*** *$S_{ideal} = S_{ideal}(m)$ if the following hold for every $m \in \mathbb{N}$:*

- *(Seed length) A sequence of inputs $(x, y) \in \mathcal{X}^{R_m} \times \mathcal{Y}^{R_m}$ to the devices can be sampled according to π_m using at most m uniformly random bits,*
- *(Completeness) If the devices behave as prescribed in the ideal strategy S_{ideal},*[3] *then*

$$\Pr(T_m(X, Y, A, B) = 1) \geq c(m), \tag{1}$$

where A, B are random variables corresponding to each device's outputs, and the probability is over $(X, Y) \sim \pi_m$ and the randomness inherent in the strategy.

[3] We refer to devices implementing the ideal strategy as *ideal devices*.

– *(Soundness) For all quantum (resp. non-signaling) strategies S for the devices in P_m, if playing according to S guarantees* $\Pr(T_m(X,Y,A,B)=1) \geq s(m)$*, then*

$$H_\infty^\varepsilon(A,B \mid T_m(X,Y,A,B)=1) \geq g(m).$$

We further elaborate on the completeness and soundness conditions. We say that the completeness of a randomness amplifier P holds *with quantum (resp. non-signaling) devices* whenever the ideal strategy can be implemented using quantum (resp. non-signaling) devices. Similarly, we say that the soundness of P holds *against quantum (resp. non-signaling) devices* if the universal quantifier in the soundness condition is over all quantum (resp. non-signaling) strategies. Generally, a stronger condition on the soundness (i.e. soundness against non-signaling devices) will imply weaker parameters, such as smaller expansion.

We note that the amount of randomness produced is measured according to its (ε-smooth) min-entropy. Motivation for this particular measure comes from the fact that it tightly characterizes the number of (ε-close to) uniform bits that can be extracted from the devices' outputs using a procedure known as an extractor (we refer to [17] for more details on using extractors for privacy amplification, including in the quantum setting). This procedure requires the use of an additional short seed of uniformly random bits, which we do not take into account here: our goal is simply to produce *entropy*, and one could in principle replace the min-entropy by, say, the Shannon entropy in the definition. It is known that randomness extraction for min-entropy sources requires an independent seed of logarithmic length [16], thus trivially limiting many protocols to exponential expansion! Since our interest is in exploring the limits and possibilities of randomness expansion – including the possibility of super-exponential expansion – we make the choice of measuring the output randomness by its min-entropy.

It may be useful to keep typical ranges for the different parameters in mind. The "asymptotic" quantity is the seed length m. Completeness will often be exponentially close to 1 in the number of rounds R, itself a function of m that can range from linear to doubly exponential (or more). The soundness and smoothness will be exponentially small in m.

We now define restricted classes of protocols which capture most of the protocols so far introduced in the literature. The definitions are extended to randomness amplifiers in the natural way.

Natural Protocols. We will say that a protocol P is *natural* if there is a two-player game G such that the ideal strategy for P is the strategy $S_G^{\otimes R}$ consisting of playing each of the R rounds of P according to an optimal (quantum or non-signaling depending on the context) strategy S_G for the game G. We say that G is the game that *underlies* P. All randomness amplifiers to date are natural according to this definition. In this paper we only consider natural protocols.

Definition 2. *Let G be a two-player game. A test function $T : \mathcal{X}^R \times \mathcal{Y}^R \times \mathcal{A}^R \times \mathcal{B}^R \to \{0,1\}$ is a* ***product test*** *with respect to G iff there exists a function $g : \{0,1\}^R \to \{0,1\}$ such that $T(x,y,a,b) = g\left(G(x_1,y_1,a_1,b_1),\ldots,G(x_R,y_R,a_R,b_R)\right)$.*

Product Protocols. We will say that a protocol P is a *product protocol* if the referee's test T is a product test with respect to some two-player game G. Intuitively, the protocol P consists of R independent instances of the game G, played in sequence (though the input distribution may not necessarily be the product distribution $\pi_G^{\otimes R}$). The referee's test is to apply a function g on the sequence of wins and losses of the devices. Natural examples of functions g for this purpose include the AND function and threshold functions, e.g. $g(w) = 1$ iff the Hamming weight of $w \in \{0,1\}^R$ is greater than $(\omega_q(G) - \eta)R$. An example of a *non*-product test would be one where, say, the referee checks that the devices output $(0,0)$ (for a given input pair) in $\frac{1}{2} \pm \epsilon$ fraction of the rounds.

Robust Protocols. Informally, a protocol is robust if small deviations from an ideal strategy are still accepted with high probability by the referee. We now provide a formal definition for such protocols. First, we introduce the notion of closeness of strategies. Let P be an R-round protocol. Let X, Y be random variables on $\mathcal{X}^R$, $\mathcal{Y}^R$ respectively distributed according to the protocol's input distribution π_P. For any strategy S, let $S_i(X_{\leq i}, Y_{\leq i})$ denote the random variable distributed as the devices' outputs in round i, conditioned on having played according to S on the input sequence $(X_{\leq i}, Y_{\leq i})$. Then we say that two strategies S and $\widehat{S}$ are η-close if for all rounds $i \in [R]$, $\left\| S_i(X_{\leq i}, Y_{\leq i}) - \widehat{S}_i(X_{\leq i}, Y_{\leq i}) \right\|_1 \leq \eta$.

Let P be a protocol with some specified ideal strategy S_{ideal} that is accepted with probability at least c in the protocol (as is when P is a member of a randomness amplifier, for example). Let T be the referee's test in the protocol. We say that P is *η-robust* if whenever the devices' strategy S for the protocol P is η-close to S_{ideal}, it holds that $\Pr(T(X, Y, A, B) = 1) \geq c$ (under strategy S). We note that this definition captures the concept of robustness against not only, say, i.i.d. noise, but also against physically plausible sources of imperfection such as misaligned mirrors, imperfect detectors, etc.

4 Lower Bounds

Let G be a two-player game in which inputs to Alice (resp. Bob) are chosen from sets $\mathcal{X}$ (resp. $\mathcal{Y}$), and answers expected in sets $\mathcal{A}$ (resp. $\mathcal{B}$). Let π be the referee's distribution on input pairs in G.

Definition 3. *We say that a two-player game G is $(p_0, \eta, 1-\xi)$-randomness generating against quantum (resp. non-signaling) players if there exists an input $x_0 \in \mathcal{X}$ such that the marginal probability $\pi(x_0) \geq p_0$ and any quantum (resp. non-signaling) strategy for the players that has success at least $\omega_q(G) - \eta$ (resp. $\omega_{ns}(G) - \eta$) satisfies* $\max_{a \in \mathcal{A}} p\big(A = a \mid X = x_0\big) \leq 1 - \xi$.

We note that for any given game G, x_0 and $\eta > 0$ the problem of approximating the smallest possible ξ such that G is $(\pi(x_0), \eta, \xi)$-randomness generating against quantum (resp. non-signaling) devices is an optimization problem for which upper bounds can be obtained through a hierarchy of semidefinite programs [10,14] (resp. a linear program). If G is an XOR game, the hierarchy converges at the first level: there is an exact semidefinite program of size polynomial

in $|\mathcal{X}||\mathcal{Y}|$. For the special case of the CHSH game, choosing $x_0 = 0$ it is known that CHSH is $(1/2, \eta, 1/2 + \sqrt{3\eta})$-randomness generating (please see the full version of the paper for details). Clearly, the condition that $\eta < \omega_q(G) - \omega_c(G)$ (resp. $\eta < \omega_{ns}(G) - \omega_c(G)$) is necessary for the game G to be randomness generating for any $\xi > 0$.

4.1 Unbounded Randomness Expansion

For any game G with input distribution π, $\varepsilon > 0$ and function $R : \mathbb{N} \to \mathbb{N}$, we introduce a simple randomness amplifier that achieves unbounded expansion, with the strong limitation that soundness only holds against devices that are restricted to play each round of the protocol in a completely isolated, though not necessarily identical, manner (in particular, the devices are memory-less but may be aware of the round number). Fix an optimal strategy S for G. Our randomness amplifier is given by the family of protocols (P_m), where protocol P_m is defined as follows.

P_m has $R = R(m)$ rounds. The rounds are divided into $(1/\varepsilon)$ blocks B_j of εR rounds each. For each block, the referee chooses a random pair of inputs $(x, y) \sim \pi$ that is used in every round of the block. The referee then checks that in every block at least a $\omega_q(G|S, x, y) - \eta$ fraction of the rounds have been won, where here $\omega_q(G|S, x, y)$ is defined as the probability that the players satisfy the game condition, conditioned on their inputs being (x, y), in the fixed strategy S (so that $\sum_{x,y} \pi(x, y)\omega_q(G|S, x, y) = \omega_q(G)$). (In the non-signaling case, replace ω_q by ω_{ns}.) The referee accepts the devices if and only if this condition holds in every block. Note that P is a non-adaptive protocol with ideal strategy $S^{\otimes R}$, completeness that goes exponentially fast to 1 with R, and seed length $O(\varepsilon^{-1})$ (where we treat the size of G as a constant).

The following lemma shows that the randomness amplifier (P_m) has good soundness and expansion that's linear in the number of rounds. Since the seed length remains a constant as $R(m)$ grows, the protocol can be used to achieve unbounded expansion. A proof can be found in the full version of the paper [9].

Lemma 1. *Let $\eta, \xi > 0$ and G a $(p_0, 4\eta, 1 - \xi)$-randomness generating game against quantum (resp. non-signaling) players. Then, for all $\varepsilon > 0$ and functions $R : \mathbb{N} \to \mathbb{N}$ the above-described randomness amplifier (P_m) has: (1) Seed length $O(\varepsilon^{-1})$, (2) Completeness $1 - e^{-\Omega(\varepsilon R(m))}$ with quantum (resp. non-signaling) devices, (3) Soundness $e^{-\Omega(1/\varepsilon)}$ against* independent *quantum (resp. non-signaling) devices, (4) Smoothness $e^{-\Omega(1/\varepsilon)}$, and (5) Expansion $g(m) = \alpha R(m)$, where α is a positive constant depending only on ξ and η. Furthermore, P is η-robust.*

4.2 Exponential Randomness Expansion

It is much more realistic to assume that the devices *do* have memory, and we analyze this case for the remainder of the section. For any game G that is randomness generating we show that there exists a corresponding randomness amplifier with exponential expansion. For simplicity we only consider quantum strategies; the

non-signaling setting is completely analogous. We introduce a randomness amplifier (P_m) which is parametrized by a randomness generating game G, a fixed set of inputs $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$, an error tolerance $\eta > 0$, a precision $\varepsilon = \varepsilon(m)$, a "checking probability" $p_c = p_c(m)$ and a number of rounds $R = R(m)$.

Fix an $m \in \mathbb{N}$. The protocol P_m proceeds as follows. The referee first samples a string $u \in \{0,1\}^R$ from a distribution $\hat{q}$ that is within statistical distance $O(\varepsilon^2)$ from the distribution q on $\{0,1\}^R$ with density $q(x) = \prod_{i\in[R]} p_c^{x_i}(1-p_c)^{1-x_i}$. In particular, $\hat{q}$ is chosen so that the referee can sample from $\hat{q}$ using only $O(p_c R \log(R))$ uniformly random bits (we refer to the full version [9] for details). He then selects inputs for the devices in the R rounds. If $u_i = 1$ inputs are selected as prescribed in G; such rounds are called "game rounds". If $u_i = 0$ they are set to the default value (x_0, y_0). Once inputs to the R rounds have been computed, the referee sequentially provides them to the devices, who produce a corresponding sequence of outputs. The referee computes the average number of rounds in which the input/output pairs satisfy the game condition G, and accepts if and only if it is at least $\omega_q(G) - \eta$. We note that P_m is a *natural*, *product* protocol for which we define the ideal strategy to consist of playing each round independently according to an optimal quantum strategy for the game G. With that ideal strategy, the protocol is also η-robust.

The following theorem shows that for any game G that is $(p_0, \eta, 1-\xi)$-randomness generating against quantum adversaries,[4] for some $\xi > 0$, the protocols (P_m) form a randomness amplifier with expansion that's linear in the number of rounds.

Theorem 3. *Let G be $(p_0, 4\eta/p_0, 1-\xi)$-randomness generating against quantum players, with input distribution π. Let m_π be the number of uniform random bits required to sample a pair of inputs $(x, y) \sim \pi$. Let $p_c, R, \varepsilon, s : \mathbb{N} \to \mathbb{N}$ be non-negative functions such that $p_c(m)R(m)(\log R(m) + m_\pi) \leq m/C$, $\varepsilon(m) \leq s(m)$, and $s(m)\varepsilon(m) > e^{-C\min(\eta^2, p_0\xi^2)p_c(m)R(m)}$ for all m, where C is a universal constant. Then the family of protocols (P_m) (as defined above), based on game G, inputs (x_0, y_0), error tolerance $(p_0\eta/4)$, precision ε, checking probability p_c and number of rounds R is a randomness amplifier with (1) Seed length m, (2) Completeness $c \geq 1 - e^{-\eta^2 R(m)}$ with quantum devices, (3) Soundness s against quantum devices, (4) Smoothness ε, and (5) Expansion $g(m) \geq \xi R(m)/5$. Furthermore, (P_m) is δ-robust for any $\delta < p_0\eta/4$.*

For any small constant $\eta > 0$, integer m and desired soundness and smoothness $\varepsilon = s$, setting $R(m) = C'm/\log(1/\varepsilon)$ and $p_c = C''\log(1/\varepsilon)/R$ for small enough C' and large enough C'' (depending on η, p_0 and ξ) will lead to parameters that satisfy the theorems' assumptions, thus guaranteeing an amount of min-entropy generated that is exponential in m for constant ε. A full proof of Theorem 3 can be found in [9].

[4] For simplicity we focus here on establishing completeness and soundness for quantum devices, but our arguments can easily be extended to the non-signaling case.

5 Upper Bounds

In this section we prove upper bounds on the expansion attainable by a wide class of randomness amplifiers. The upper bounds are proved by exhibiting "cheating strategies" for the two devices D_A and D_B that fool a referee into accepting, while producing an amount of entropy that is at most doubly exponential in the referee's seed length. In particular, our bounds on output entropy are independent of the number of rounds.

The main idea behind the cheating strategies we exhibit is that, after a sufficiently large number of rounds, there are inevitable correlations between the referee's inputs to the devices that hold irrespective of the referee's choice of random seed. These correlations can be inferred from the given input distribution π of the protocol, before it begins. In Theorems 4 and 5 we use the observation that after a number of rounds that is doubly exponential in the referee's seed length, the inputs to D_A and D_B in the current round i must be identical to their inputs in some previous round $j < i$. If the referee's test is particularly simple (as it is assumed to be in Theorem 4), then the devices can pass the protocol by simply copying their answers from round j. More generally, we show that for robust protocols there will be a set of rounds $J \subseteq [R]$ such that $|J| = 2^{O(2^m)}$ (where m is the referee's seed length), and a strategy for the devices to deterministically recombine their respective answers from the rounds in J into answers for the rounds in $[R]\backslash J$. It follows that the devices' output entropy is at most $O(|J|) = 2^{O(2^m)}$.

An important element of the cheating strategies we present is the **input matrix**, which is defined for any nonadaptive protocol as follows.

Definition 4 (Input matrix). *Let P be an R-round, non-adaptive protocol with seed length m. The input matrix M_P is the $R \times 2^m$ matrix whose (i, σ)-entry is $M_P(i, \sigma) = (X(\sigma)_i, Y(\sigma)_i)$, where here $X(\sigma)$ (resp. $Y(\sigma)$) are the input sequences for device D_A (resp. D_B) chosen by the referee on seed $\sigma \in \{0, 1\}^m$.*

Let $M = M_P$ denote the input matrix for some protocol P. We let $M_i \in (\mathcal{X} \times \mathcal{Y})^{2^m}$ denote the ith row of an input matrix $M = M_P$. We define the set $F(M) \subseteq [R]$ as the set of round indices i such that $i \in F(M)$ iff $M_i \neq M_j$ for all $j < i$. A simple, but crucial observation we use in our upper bounds is the following: Let P be a protocol with seed length m and input alphabets $\mathcal{X}, \mathcal{Y}$. Then $|F(M)| \leq |\mathcal{X} \times \mathcal{Y}|^{2^m}$.

5.1 A Simple Doubly Exponential Bound

We first demonstrate a doubly exponential upper bound on randomness amplifiers that are based on perfect games, which are games G such that $\omega_q(G) = 1$ (or $\omega_{ns}(G) = 1$, if we're allowing devices with full non-signaling power). In these protocols, the referee checks that the devices win every single round.

Theorem 4. *Let G be such that $\omega_q(G) = 1$ (resp. $\omega_{ns}(G) = 1$). Let $P = (P_m)$ be a randomness amplifier with input (resp. output) alphabets $\mathcal{X}, \mathcal{Y}$ (resp. $\mathcal{A}, \mathcal{B}$)*

and in which the referee's test consists in verifying that the devices win G in every round. Suppose completeness and soundness of P both hold with quantum (resp. non-signaling) devices. Then the expansion of P satisfies $g(m) \leq |\mathcal{X} \times \mathcal{Y}|^{2^m} \log|\mathcal{A} \times \mathcal{B}| - \log(1 - \varepsilon(m))$, where $\varepsilon(m)$ is the smoothness of P.

We only sketch the proof here; we give a more general argument in the next section. The idea of the proof is as follows: in each round i, the devices check whether $i \in F(M)$ or not, where $M = M_{P_m}$ is the input matrix corresponding to protocol P_m. If it is, then the devices play according to the ideal, honest strategy that wins G with probability 1. If not, then there must exist a $j \in F(M)$, $j < i$, such that $M_i = M_j$. Thus, regardless of the referee's seed, it must be that $(x_i, y_i) = (x_j, y_j)$ always. In that case, the devices will simply replay their outputs (a_j, b_j) from that round, independently setting $a_i := a_j$ and $b_i := b_j$. Since we can assume that round j was won with probability 1, round i must be won with probability 1 as well. It is easy to see that the only entropy-generating rounds are those in $F(M)$, and the theorem follows from bound on $|F(M)|$ described above.

5.2 A Doubly Exponential Bound for Robust Protocols

In this section we generalize the bound from the previous section to show a doubly exponential upper bound on the expansion achievable by any randomness amplifier based on a protocol that is non-adaptive and robust. In particular, the underlying game G may not be perfectly winnable, and the referee's test T may not necessarily check that the devices win G in every single round. The fact that we allow an arbitrary test T in the protocol complicates the proof, as the referee may now for example check for obvious answer repetitions in the players' answers to identical question pairs, and thereby easily detect cheating strategies of the form described in Section 5.1. Nevertheless, we will design a somewhat more elaborate cheating strategy for the devices in any such protocol, that prevents it from achieving unbounded expansion.

Theorem 5. *Let $P = (P_m)$ be a natural, η-robust randomness amplifier such that completeness and soundness both hold with respect to quantum (resp. non-signaling) devices. Let $K_m = \Omega\left(\frac{1}{\eta^2} \log \frac{|\mathcal{A}\times\mathcal{B}|\cdot|F(M_{P_m})|}{\eta}\right)$. Then the expansion of P satisfies $g(m) \leq K_m \cdot |F(M_{P_m})| \cdot \log|\mathcal{A} \times \mathcal{B}| - \log(1 - \varepsilon(m))$, where $\mathcal{A}, \mathcal{B}$ are the output alphabets of P, and $\varepsilon(m)$ is the smoothness of P.*

Combined with the bound on $|F(M)|$, the theorem implies that any η-robust randomness amplifier P must have a expansion $g(m) = 2^{O(2^m)}$ (where the constant in the $O(\cdot)$ depends only on η, the smoothness ε, and the alphabets $\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}$). This in particular demonstrates that unbounded randomness expansion as demonstrated in Lemma 1 is impossible as soon as the devices are allowed to have (classical) memory.

For space considerations we omit the proof; we sketch the idea here (we refer to the full version of the paper [9] for details). Instead of directly reusing outputs corresponding to identical pairs of inputs, as described in Section 5.1, the

devices first repeatedly apply the protocol's ideal quantum (resp. non-signaling) strategy for game G in order to locally generate a discrete approximation to the corresponding distribution on outputs. Whenever they receive a pair of questions for which they already computed such an approximation, they use shared randomness to jointly sample a pair of answers from the approximating distribution. To conclude we use the probabilistic method to derandomize the shared sampling step (which would otherwise still lead to the generation of a constant amount of entropy per round).

5.3 An Exponential Upper Bound for Protocols with Non-signaling Devices

In this section we demonstrate exponential upper bounds on the attainable expansion of a class of non-adaptive randomness amplifiers for which completeness holds with respect to non-signaling devices. We address protocols using the CHSH game, which have been widely studied in the literature [14,19].

Theorem 6. *Let $P = (P_m)$ be a randomness amplifier in which completeness and soundness both hold with non-signaling devices, and for each m the referee's test T_m is a product test with respect to the* CHSH *game. Then $g(m) \leq 2^{2m+2} - \log(1 - \varepsilon(m))$, where $\varepsilon(m)$ is the smoothness parameter of P.*

For space considerations, we omit the proof; we refer to the full version of the paper [9] for details.

Theorem 6 exhibits a scenario in which the specific structure of the underlying game G and the protocol can be used to give an exponential improvement over Theorem 4. For simplicity we have constrained the theorem statement to protocols involving the CHSH game, but the proof can be extended to establish the same result when G is a balanced 2-player XOR game, as well as the (3-player) GHZ game, which has played an important role in early randomness expansion results [7]. We refer to the full version of the paper for additional details.

We remark that Theorem 6 implies a "meta-theorem" that says that the type of analysis performed in [19] cannot be improved to have more than exponential expansion. Any randomness amplifier based on the CHSH game in which the referee only checks that the devices won more than a certain fraction of the rounds, and where the analysis of soundness only uses the fact that the devices are non-signaling, by Theorem 6, must be limited to exponential expansion. The randomness amplifier in [19] is of this form, and hence modifying it to obtain super-exponential expansion would require either a non-product test, *or* an analysis that uses the fact that the devices can "only" be quantum!

Acknowledgments. T.V. thanks Andy Drucker and Avi Wigderson for discussions. M.C. and H.Y. thank Scott Aaronson for his engaging class on Quantum Complexity Theory, for which some of the upper bounds were developed as a course project. H.Y. also thanks Joseph Bebel for helpful comments. We thank the anonymous RANDOM referees as well as Marco Tomamichel for comments that helped improve the presentation of our paper.

References

1. Aravind, P.K.: The magic squares and Bell's theorem. Tech. rep., arXiv:quant-ph/0206070 (2002)
2. Aspect, A., Grangier, P., Roger, G.: Experimental tests of realistic local theories via Bell's theorem. Phys. Rev. Lett. 47(7), 460–463 (1981)
3. Brunner, N., Cavalcanti, D., Pironio, S., Scarani, V., Wehner, S.: Bell nonlocality. Tech. rep., arXiv:1303.2849 (2013)
4. Cirel'son, B.: Quantum generalizations of Bell's inequality. Letters in Mathematical Physics 4(2), 93–100 (1980)
5. Cleve, R., Høyer, P., Toner, B., Watrous, J.: Consequences and limits of nonlocal strategies. In: Proc. 19th IEEE Conf. on Computational Complexity (CCC 2004), pp. 236–249. IEEE Computer Society (2004)
6. Colbeck, R.: Quantum and Relativistic Protocols for Secure Multi-Party Computation. Ph.D. thesis, Trinity College, University of Cambridge (November 2006)
7. Colbeck, R., Kent, A.: Private randomness expansion with untrusted devices. Journal of Physics A: Mathematical and Theoretical 44(9), 095305 (2011)
8. Colbeck, R., Renner, R.: Free randomness can be amplified. Nature Physics 8(6), 450–454 (2012)
9. Coudron, M., Vidick, T., Yuen, H.: Robust randomness amplifiers: Upper and lower bounds. Tech. rep., arXiv:1305.6626 (2013)
10. Doherty, A.C., Liang, Y.C., Toner, B., Wehner, S.: The quantum moment problem and bounds on entangled multi-prover games. In: Proc. 23rd IEEE Conf. on Computational Complexity (CCC 2008), pp. 199–210. IEEE Computer Society (2008)
11. Fehr, S., Gelles, R., Schaffner, C.: Security and composability of randomness expansion from Bell inequalities. Phys. Rev. A 87, 012335 (2013), `http://link.aps.org/doi/10.1103/PhysRevA.87.012335`
12. Linden, N., Popescu, S., Short, A.J., Winter, A.: Quantum nonlocality and beyond: Limits from nonlocal computation. Phys. Rev. Lett. 99, 180502 (2007)
13. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Cambridge University Press (2010)
14. Pironio, S., Acin, A., Massar, S., De La Giroday, A.B., Matsukevich, D.N., Maunz, P., Olmschenk, S., Hayes, D., Luo, L., Manning, T.A., et al.: Random numbers certified by Bell's theorem. Nature 464(7291), 10 (2010)
15. Pironio, S., Massar, S.: Security of practical private randomness generation. Phys. Rev. A 87, 012336 (2013), `http://link.aps.org/doi/10.1103/PhysRevA.87.012336`
16. Radhakrishnan, J., Ta-Shma, A.: Bounds for dispersers, extractors, and depth-two superconcentrators. SIAM Journal on Discrete Mathematics 13(1), 2–24 (2000)
17. Renner, R.: Security of Quantum Key Distribution. Ph.D. thesis, Swiss Federal Institute of Technology Zurich (September 2005)
18. Shaltiel, R.: Recent developments in explicit constructions of extractors. Bulletin of the European Association for Theoretical Computer Science 77, 67–95 (2002)
19. Vazirani, U., Vidick, T.: Certifiable quantum dice: or, true random number generation secure against quantum adversaries. In: Proceedings of the 44th Symposium on Theory of Computing, STOC 2012, pp. 61–76. ACM (2012)

The Power of Choice for Random Satisfiability

Varsha Dani[1], Josep Diaz[2,⋆], Thomas Hayes[1,⋆⋆], and Cristopher Moore[3,⋆⋆⋆]

[1] University of New Mexico
[2] Universitat Politècnica de Catalunya
[3] Santa Fe Institute

Abstract. We consider Achlioptas processes for k-SAT formulas. That is, we consider semi-random formulas with n variables and $m = \alpha n$ clauses, where each clause is a choice, made on-line, between two or more independent and uniformly random clauses. Our goal is to move the sat/unsat transition, making the density $\alpha = m/n$ at which these formulas become unsatisfiable larger or smaller than the satisfiability threshold α_k for uniformly random k-SAT formulas. We show that three choices suffice to raise the threshold for any $k \geq 3$, and that two choices suffice for $3 \leq k \leq 50$. We also show that (assuming the threshold conjecture is true) two choices suffice to lower the threshold for all $k \geq 3$, and that (unconditionally) a constant number of choices suffice.

1 Introduction

The Erdős-Rényi model of random graphs undergoes a celebrated phase transition. Specifically, suppose we form a random graph $G(n, m)$ with n vertices and m edges by choosing m times uniformly from the $\binom{n}{2}$ possible edges. The average degree of this graph is $d = 2m/n$. If $d < 1$, then with high probability in the limit $n \to \infty$, $G(n, m)$ consists almost entirely of trees, and the largest component has size $O(\log n)$. But if $d > 1$, then with high probability $G(n, m)$ has a giant connected component containing $\Theta(n)$ vertices.

In 2001, Dimitris Achlioptas posed the following question. Suppose at each step we are presented with *two* uniformly random edges. We are allowed to choose between them, adding one of them to the graph and throwing away the other. We play this game on-line; that is, our choice can depend on the graph up to this point, but not on future pairs of edges. Can we delay the appearance of the giant component, ensuring that the largest component has size $o(n)$ after $m = cn$ edges for some $c > 1/2$?

A positive answer was given by Bohman and Frieze [5], who showed that two choices suffice to delay the giant up to $c = 0.535$. Achlioptas, D'Souza, and Spencer [1] studied a particular rule where we choose the edge that minimizes the product of the component sizes of its endpoints; this exhibits a phenomenon

⋆ Partially supported by the CYCIT: TIN2007-66523 (FORMALISM).
⋆⋆ Partially supported by NSF CAREER Award CCF-1150281 and NSF grant CCF-1219117.
⋆⋆⋆ Partially supported by NSF grant CCF-1219117.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 484–496, 2013.

they call *explosive percolation* (although Riordan and Warnke [28] showed that the transition is continuous). It is also possible to speed up the appearance of the giant component [14,6]; Spencer and Wormald [30] showed that it can be brought into existence at $c = 0.334$.

In analogy with $G(n, m)$, we can consider random k-SAT formulas $F_k(n, m)$. Specifically, given n variables $x_1, \ldots, x_n$, we create a k-SAT formula by choosing m clauses independently and uniformly from the $2^k\binom{n}{k}$ possibilities. The *satisfiability threshold conjecture* states that there is a critical density $\alpha_k = m/n$ at which $F_k(n, m)$ undergoes a phase transition from satisfiable to unsatisfiable:

Conjecture 1 *For each $k \geq 2$, there is a constant α_k such that*

$$\lim_{n\to\infty} \Pr[F_k(n, \alpha n) \text{ is satisfiable}] = \begin{cases} 1 & \alpha < \alpha_k \\ 0 & \alpha > \alpha_k \,. \end{cases}$$

This conjecture has been proved only for $k = 2$ [9,16,13], where $\alpha_2 = 1$. For the NP-complete case $k \geq 3$, Friedgut [15] showed the existence of nonuniform thresholds $\alpha_k(n)$ but it is not known whether these converge to a constant as n goes to infinity. Nevertheless, there are strong arguments from statistical physics that the conjecture is true, and very precise estimates for the value of α_k from calculations using the cavity method (see [22,23] for reviews).

There are rigorous upper and lower bounds on α_k assuming it exists. That is, there are known values $\alpha_k^{\text{low}}, \alpha_k^{\text{high}}$ such that w.h.p. $F_k(n, \alpha n)$ is satisfiable if $\alpha < \alpha_k^{\text{low}}$ and unsatisfiable if $\alpha > \alpha_k^{\text{high}}$. In that case, we write $\alpha_k^{\text{low}} \leq \alpha_k \leq \alpha_k^{\text{high}}$. Specifically, for $k = 3$ we have [11,17,19]

$$3.52 \leq \alpha_3 \leq 4.4898 \,, \tag{1}$$

while the cavity method gives $\alpha_3 = 4.267$. For large k, the best known rigorous bounds on the threshold [10,20] behave as

$$2^k \ln 2 - \frac{3 \ln 2}{2} - o(1) \leq \alpha_k < 2^k \ln 2 - \frac{1 + \ln 2}{2} \,, \tag{2}$$

where $o(1)$ tends to zero as k tends to infinity. The cavity method of statistical physics [21] gives

$$\alpha_k = 2^k \ln 2 - \frac{1 + \ln 2}{2} + O(2^{-k}) \,.$$

Sinclair and Vilenchik [29] asked whether Achlioptas processes can delay the satisfiability/unsatisfiability transition for k-SAT. In other words, suppose at each step we are given a choice of two independent and uniformly random clauses. We choose one of them and add it to the formula, and our goal is keeping the formula satisfiable up to $m = \alpha n$ clauses for some $\alpha > \alpha_k$. They showed that two choices are enough to delay the 2-SAT transition up to $\alpha = 1.0002$, and also that two choices can delay the k-SAT transition for $k = \omega(\log n)$. Perkins [27] shows that, for each $k \geq 3$, there exists a finite τ such that τ choices can delay the k-SAT transition, and that 5 choices suffice for $k = 3$. Furthermore, it follows

from his Lemma 1 and known upper bounds on the k-SAT threshold that 7 choices suffice for all k, and that 3 choices suffice for all sufficiently large k.

We improve these results in the following ways. First, we give a simple, nonadaptive strategy that, given a choice between three clauses, increases the k-SAT threshold for all $k \geq 2$. Secondly, we give a two-choice strategy that increases the threshold for all $3 \leq k \leq 50$, and we conjecture that it works for all large k as well. Finally, we give a simple strategy that, assuming the threshold conjecture, lowers the k-SAT threshold for all k using two choices. Without this assumption, the same method still works for two choices for sufficiently large k, but may require a larger number of choices to lower the threshold for small values of k.

2 Three Choices Suffice to Raise the Threshold for All k

In this section and the next, we show that a constant number of choices suffice to raise the satisfiability threshold. Our strategy is simple and nonadaptive. Indeed, it is oblivious to the "topology" of the formula—which variables appear together in clauses—and is sensitive only to the signs of the literals. Given a choice of τ clauses, we choose the one with the largest number of positive literals.

To show that the resulting k-SAT formula is satisfiable, we convert it into an ℓ-SAT formula in the following way: for each k-SAT clause c, we form an ℓ-SAT clause by taking ℓ of the most positive literals in c. If the resulting ℓ-SAT formula is satisfiable, then so is the original k-SAT formula. In Theorem 1, we use $\ell = 2$; in Theorems 2–4, we use $\ell = 3$.

We note that Perkins [27] used a similar strategy, with $\ell = 2$, to show that a constant number of choices suffice for any k. Specifically, given τ choices, if any of the first $\tau - 1$ clauses have two or more positive literals, we choose one of them; otherwise, we choose the τth clause. Our rule is slightly more selective, since even if none of the clauses have two positive literals, it chooses a clause with one positive literal if one exists.

Theorem 1. *Three choices suffice to increase the k-SAT threshold for $k \geq 2$.*

Proof. As described above, we simply take the clause c with the most positive literals. We then generate a 2-SAT formula by taking two of the most positive literals from each clause. Specifically, if c has two or more positive literals, we form a 2-SAT clause by choosing uniformly from all such pairs; if c has exactly one positive literal, we take it and choose uniformly from the $k - 1$ others; and if all of c's literals are negative, we choose uniformly from all $\binom{k}{2}$ pairs.

If c is the most-positive of τ uniformly random clauses, then the probabilities that the resulting 2-SAT clause has 0, 1, or 2 positive literals are

$$\begin{aligned} p_0 &= 2^{-k\tau} \\ p_1 &= \left(2^{-k}(k+1)\right)^{\tau} - p_0 \\ p_2 &= 1 - p_0 - p_1 \,. \end{aligned} \tag{3}$$

If there are $m = \alpha n$ clauses, this gives a biased random 2-SAT formula with, in expectation, $\alpha p_0 n$, $\alpha p_1 n$, and $\alpha p_2 n$ clauses of these three types. Note that the variables appearing in each clause are independent and uniformly random.

Recall that a 2-SAT formula on n variables is equivalent to a directed graph on $2n$ vertices, corresponding to the literals x_i and $\overline{x}_i$ for each $1 \le i \le n$. Each clause $(x_i \vee x_j)$ is equivalent to a pair of edges, namely the implications $\overline{x}_i \to x_j$ and $\overline{x}_j \to x_i$. The formula is satisfiable if and only if no contradictory cycle exists, leading from x_i to $\overline{x}_i$ and back to x_i for some i.

Unit clause propagation is the process of satisfying a unit clause, i.e. a clause consisting of a single literal, and generating the unit clauses implied by it and whatever 2-clauses that variable appears in. For instance, if $(\overline{x}_i \vee x_j)$ is one of the 2-clauses in the formula, satisfying the unit clause (x_i) will generate the unit clause (x_j). Consider the positive unit clause (x_i). Satisfying it will create new unit clauses from each of the 2-clauses that contain the literal $\overline{x}_i$. In expectation, there are $2\alpha p_0$ of these of the type $(\overline{x}_i \vee \overline{x}_j)$ and αp_1 of type $(\overline{x}_i \vee x_j)$. The former give rise to negative unit clauses $(\overline{x}_j)$, while the former give rise to positive unit clauses (x_j). Similarly, a negative unit clause $(\overline{x}_i)$ will give rise to on average αp_1 negative unit clauses and $2\alpha p_2$ positive unit clauses, from the 2-clauses containing the literal x_i. Unit clause propagation is thus described by a two-type branching process, with a matrix αM where

$$M = \begin{pmatrix} p_1 & 2p_2 \\ 2p_0 & p_1 \end{pmatrix}, \tag{4}$$

where we treat the number of negative and positive unit clauses in the current generation as a column vector and multiply by M on the left.

Given an initial vector u of unit clauses, the expected population generated by the entire process is

$$\left(\mathbb{1} + \alpha M + (\alpha M)^2 + \cdots\right) \cdot u\,.$$

If the largest eigenvalue λ of M obeys $\alpha\lambda < 1$, this converges to $(\mathbb{1} - \alpha M)^{-1} \cdot u$, and in expectation just $O(1)$ unit clauses are implied by the initial one. Intuitively, this makes it very unlikely that a contradictory loop of implications exists, and suggests that the 2-SAT formula is satisfiable with high probability.

Indeed, this was proved by Mossel and Sen [26]. They showed that the critical threshold for random 2-SAT formulas of this kind is exactly

$$\alpha^* = \frac{1}{\lambda} = \frac{1}{p_1 + 2\sqrt{p_0 p_2}}\,.$$

For the unbiased case $p_1 = 1/2$ and $p_0 = p_2 = 1/4$, this reproduces the 2-SAT threshold $\alpha_2 = 1$. Putting in our expressions (3) for p_0, p_1, and p_2 gives

$$\alpha^* = \frac{2^{k\tau/2}}{2^{-k\tau/2}((k+1)^\tau - 1) + 2\sqrt{1 - (2^{-k}(k+1))^\tau}}$$

For large k, α^* grows as $2^{k\tau/2}/2$. If we set $\tau = 3$, then α^* exceeds the k-SAT threshold for all $k \ge 3$. In particular, for $k = 3$ we have $\alpha^* > 4.86$, which exceeds

the best known upper bound on α_3 of 4.4898 [11]. For $k \geq 4$, α^* exceeds the first moment upper bound $2^k \ln 2$. □

Note that we have shown not just that three choices are enough to generate satisfiable formulas above the satisfiability threshold, but that these formulas can be satisfied in polynomial time: just use the polynomial-time algorithm for 2-SAT to find a satisfying assignment. For the case $k = 2$ and $\tau = 2$, we have also shown that two choices raise the 2-SAT threshold to 1.203, which improves the results of [29,27].

Note also that setting $\tau = 1$ in the proof of Theorem 1 shows that the threshold for random k-SAT without any choices grows as $\alpha_k = \Omega(2^{k/2})$. This is far below the second moment lower bound $\Omega(2^k)$ [2,4,10], but the proof is much simpler.

3 Two Choices Suffice to Raise the Threshold for $3 \leq k \leq 50$

In this section we show that two choices suffice for k up to 50. We do this by analyzing simple linear-time algorithms with differential equations. Regrettably, these equations seem too complicated to solve analytically; thus we are not able to prove that these results hold for all $k \geq 3$, though we conjecture that they do.

We start by showing that a particularly simple algorithm works for $5 \leq k \leq 50$. We then use slightly more sophisticated algorithms to raise the threshold for $k = 3$ and $k = 4$.

Theorem 2. *Two choices suffice to increase the k-SAT threshold if* $5 \leq k \leq 50$.

Proof. Our strategy is the same as before: given a choice of τ clauses, take the one with the most positive literals. We then form a 3-SAT clause by choosing uniformly from among the most-positive triplets of literals. Analogous to (3), the probability that the resulting clause has 0, 1, 2, or 3 positive literals is

$$\begin{aligned} p_0 &= 2^{-k\tau} \\ p_1 &= \left(2^{-k}(k+1)\right)^\tau - p_0 \\ p_2 &= \left(2^{-k}\left(\binom{k}{2} + k + 1\right)\right)^\tau - p_1 - p_0 \\ p_3 &= 1 - p_0 - p_1 - p_2 \,. \end{aligned} \tag{5}$$

Now consider the following algorithm, which we call BUC for Biased Unit Clause. At each step it sets some variable x permanently, removing clauses satisfied by that setting, and shortening clauses that disagree with it.

1. (Forced step) If there exist unit clauses, choose one uniformly and satisfy it.
2. (Free step) Else, choose x uniformly from all unset variables, and set x true.

This is identical to the UC algorithm for random k-SAT studied by Chao and Franco [7,8] except that, on a free step, UC flips a coin to determine the truth

value of x. If at any point we have two contradictory unit clauses, we simply give up rather than backtracking. Our goal is to use differential equations to show that BUC succeeds with positive probability. The existence of a nonuniform threshold [15], which we claim applies to these biased 3-SAT formulas as well, then implies that they are satisfiable with high probability.

After T of the variables have been set, let $S_{ij}(T)$ denote the number of i-clauses with j positive literals, for $i = 2, 3$ and $0 \le j \le i$. Initially we have w.h.p. $S_{3,j}(0) = \alpha p_j n + o(n)$ and $S_{2,j}(0) = 0$. Let $q_0(T)$ and $q_1(T)$ denote the probability that the variable on the Tth step is set false or true respectively. Then the expected change in S_{ij} at each step is

$$\text{for all } 0 \le j \le 3\,, \quad \mathbb{E}\left[\Delta S_{3,j}\right] = -\frac{3S_{3,j}}{n-T} + o(1)$$

$$\text{for all } 0 \le j \le 2\,, \quad \mathbb{E}\left[\Delta S_{2,j}\right] = \frac{(3-j)q_1 S_{3,j} + (j+1)q_0 S_{3,j+1} - 2S_{2,j}}{n-T} + o(1)\,.$$

The key fact behind these equations is that, at all times throughout the algorithm's progress, the formula consisting of the remaining clauses is uniformly random once we condition on the number of clauses of each type. In particular, the variables appearing in each clause are uniformly random among the $n - T$ unset variables, as is the variable x set on a given step. Thus each 3-clause is either satisfied or shortened with probability $3/(n-T)$; if it has j positive literals and we set x false, then with probability $j/(n-T)$ it becomes a 2-clause with $j-1$ positive literals; and so on.

Rescaling to real-valued variables $t = T/n$ and $s_{ij}(t) = S_{ij}(tn)/n$ in the usual way gives the differential equations

$$\text{for all } 0 \le j \le 3\,, \quad \frac{\mathrm{d}s_{3,j}}{\mathrm{d}t} = -\frac{3s_{3,j}}{1-t} \tag{6}$$

$$\text{for all } 0 \le j \le 2\,, \quad \frac{\mathrm{d}s_{2,j}}{\mathrm{d}t} = \frac{(k-j)q_1 s_{3,j} + (j+1)q_0 s_{3,j+1} - 2s_{2,j}}{1-t}\,, \tag{7}$$

with the initial conditions $s_{3,j}(0) = \alpha p_j$ and $s_{2,j}(0) = 0$. Then classic results [32] show that, with high probability, $S_{ij}(T) = s_{ij}(T/n)n + o(n)$ for all T, where $s_{ij}(t)$ is the unique solution to this system of differential equations.

The caveat to this, of course, is that a contradictory pair of unit clauses does not appear. Standard arguments show that as long as the branching process of unit clauses stays subcritical throughout the algorithm, then the probability that no contradiction occurs, and that the algorithm succeeds in satisfying all the clauses, is $\Theta(1)$.

Analogous to (4), the unit clauses obey a two-type branching process between negative and positive unit clauses, where the expected number of children of each type is within $o(1)$ of the matrix

$$M = \frac{1}{1-t}\begin{pmatrix} s_{2,1} & 2s_{2,0} \\ 2s_{2,2} & s_{2,1} \end{pmatrix}\,. \tag{8}$$

We can group steps together into rounds, where each round consists of a free step followed by a cascade of forced steps. Let λ denote the largest eigenvalue of

M. As long as $\lambda < 1$, the branching process is subcritical, and the total expected number b_0, b_1 of variables set false or true respectively in a round is

$$\begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \left(\mathbb{1} + M + M^2 + \cdots\right) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = (\mathbb{1} - M)^{-1} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} ,$$

where we use the fact that the initial free step in each round sets a variable true. Averaging over many steps, but not so many that M changes appreciably, the probability that a variable is set false or true is

$$q_0 = \frac{b_0}{b_0 + b_1} , \; q_1 = \frac{b_1}{b_0 + b_1} .$$

Similar analyses of multi-type branching processes appear in [3,18].

Table 1. The lower bound α^*_{BUC} achieved by choosing the clause with the most positive literals, and running Biased Unit Clause on the 3-SAT formula consisting of one of the the most-positive triplets of each clause. For $5 \le k \le 50$, α^*_{BUC} exceeds the first-moment upper bound on α_k, showing that two choices suffice to raise the threshold. See Figure 1.

k	3	4	5	6	7	8	9	10
α^*_{BUC}	4.232	9.491	24.306	66.811	190.806	554.106	1610.88	4637.05
$2^k \ln 2$			22.18	44.36	88.72	177.45	354.89	709.78

The differential equation (6) for $s_{3,j}$ is easy to solve: namely,

$$s_{3,j} = \alpha p_j (1 - t)^3 .$$

We integrate (7) numerically, with guaranteed bounds on the error, and use binary search to find the largest α, up to some precision, such that $\max_t \lambda(t) < 1$. In Table 1 and Figure 1, we show the resulting lower bound α^*_{BUC} for the first few values of k. For $k = 3$ and $k = 4$, α^*_{BUC} is below the conjectured values of the threshold [21], namely 4.267 and 9.931. But for $5 \le k \le 50$, α^*_{BUC} exceeds the first moment upper bound $2^k \ln 2$. □

Asymptotically, α^*_{BUC} seems to grow roughly as β^k, where $\beta \approx 2.52$. It is tempting to think that we can prove a lower bound on α^*_{BUC} sufficient to show that two choices suffice for all $k > 50$ as well, but we have not been able to do that.

The next two theorems use slight improvements to Theorem 2 to raise the threshold for $k = 3$ and $k = 4$.

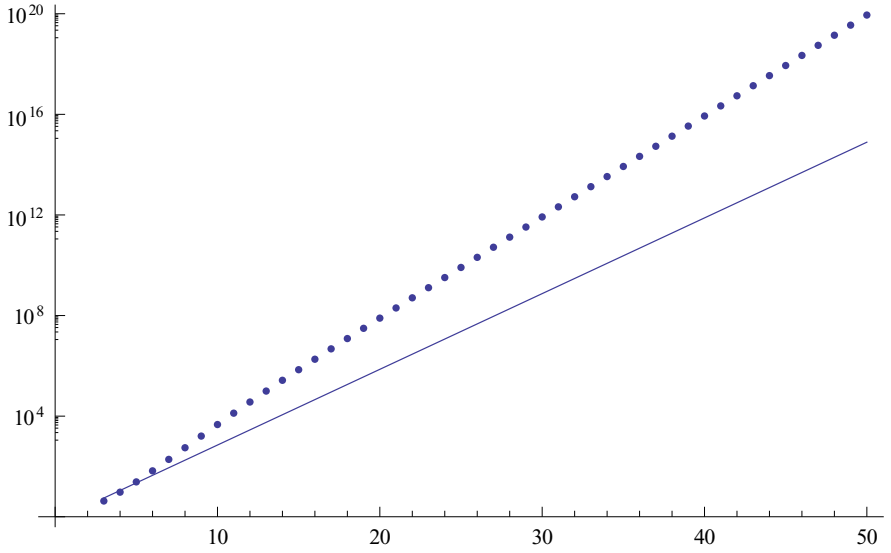


Fig. 1. Log plot of α^*_{BUC} and $2^k \ln 2$ for $3 \le k \le 50$

Theorem 3. *Two choices suffice to increase the* 4*-SAT threshold.*

Proof. Given two clauses, we again take the one with more positive clauses, but now we apply the BUC algorithm directly to the resulting 4-SAT formula. Most of the analysis of Theorem 2 goes through unchanged, except that the probability that a clause has a given number of positive literals is now

$$p_0 = \frac{1}{256}\,,\ p_1 = \frac{3}{32}\,,\ p_2 = \frac{3}{8}\,,\ p_3 = \frac{13}{32}\,,\ p_4 = \frac{31}{256}\,.$$

The differential equations (7) for the density of 2-clauses and the matrix M for the branching process of unit clauses (8) are unchanged. The differential equations for 4- and 3-clauses are now

$$\begin{aligned} &\text{for all } 0 \le j \le 4\,, \quad \frac{\mathrm{d}s_{4,j}}{\mathrm{d}t} = -\frac{4s_{4,j}}{1-t} \\ &\text{for all } 0 \le j \le 3\,, \quad \frac{\mathrm{d}s_{3,j}}{\mathrm{d}t} = \frac{(4-j)q_1 s_{4,j} + (j+1)q_0 s_{4,j+1} - 3s_{3,j}}{1-t}\,, \end{aligned} \tag{9}$$

and the initial conditions are $s_{4,j}(0) = \alpha p_j$ and $s_{3,j}(0) = s_{2,j}(0) = 0$.

Integrating this system numerically, we find that M's largest eigenvalue λ is less than 1 up to $\alpha = 10.709$. This is less than the naive first moment upper bound on α_4, but it exceeds an improved upper bound of 10.217 based on counting locally maximal assignments [12]. □

Finally, we use a biased version of the Short Clause (SC) algorithm, which Chvatal and Reed used to prove a lower bound on the 3-SAT threshold [9], to show that two choices can delay the satisfiability transition in 3-SAT.

Theorem 4. *Two choices suffice to increase the* 3*-SAT threshold.*

Proof. Once again our strategy is to take the more positive of the two clauses. The probability that a clause has a given number of positive literals is

$$p_0 = \frac{1}{64}\,,\ p_1 = \frac{15}{64}\,,\ p_2 = \frac{33}{64}\,,\ p_3 = \frac{15}{64}\,.$$

We now analyze the following algorithm, which we call Biased Short Clause (BSC).

1. (Forced step) If there exist unit clauses, choose one uniformly and satisfy it.
2. (Free step) Otherwise, if there are any 2-clauses, choose one uniformly. If it has any positive literals, choose one uniformly and satisfy it. If both its literals are negative, choose one uniformly and satisfy it.
3. (Really free step) If there are no unit clauses or 2-clauses, choose x uniformly from the unset variables and choose x's truth value uniformly.

This is identical to Short Clause [9] except that, whenever possible, we satisfy the chosen 2-clause by setting a variable true.

During the critical phase of the algorithm, there are $\Theta(n)$ 2-clauses, so we can effectively ignore the possibility of a really free step. Let p_{free} denote the probability that a given step is free. The differential equations for 3- and 2-clauses are then

$$\text{for all } 0 \le j \le 3\,, \quad \frac{\mathrm{d}s_{3,j}}{\mathrm{d}t} = -\frac{3s_{3,j}}{1-t} \tag{10}$$

$$\text{for all } 0 \le j \le 2\,, \quad \frac{\mathrm{d}s_{2,j}}{\mathrm{d}t} = \frac{(k-j)q_1 s_{3,j} + (j+1)q_0 s_{3,j+1} - 2s_{2,j}}{1-t} - p_{\text{free}}\frac{s_{2,j}}{s_{2,0}+s_{2,1}+s_{2,2}}\,, \tag{11}$$

where the additional term is due to the fact that we choose and satisfy a random 2-clause on every free step.

As before, consider a round consisting of a free step followed by a cascade of forced steps, and let b_0 and b_1 denote the total expected number of variables set false or true during a round. The probability that a given step is free is 1 divided by the expected length of the round,

$$p_{\text{free}} = \frac{1}{b_0 + b_1}\,,$$

and the probability that a given step sets a variable false or true is $q_0 = b_0/(b_0 + b_1)$ and $q_1 = b_1/(b_0 + b_1)$ respectively. The matrix M describing the branching process of unit clauses is the same as in BUC. However, the initial population of unit clauses in each round is different. Rather than always setting a variable true, a free step sets a variable true if the chosen 2-clause has at least one positive literal, and otherwise it sets a variable false. Thus

$$\begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \frac{1}{s_{2,0}+s_{2,1}+s_{2,2}}\,(\mathbb{1} - M)^{-1} \cdot \begin{pmatrix} s_{2,0} \\ s_{2,1}+s_{2,2} \end{pmatrix}\,.$$

Integrating this system numerically, we find that M's largest eigenvalue λ stays below 1 for all t as long as $\alpha < 4.581$. This exceeds the best known upper bound $\alpha_3 < 4.4898$, completing the proof. □

All these results show that two choices are enough to create a formula at a density above α_k that can be satisfied, with probability $\Theta(1)$, in linear time.

4 Two Choices Suffice to Lower the Threshold, If There Is One

We now show that two choices are enough to lower the satisfiability threshold if the threshold exists. If there is no threshold, we can still lower it; we explain below what we mean by this tongue-in-cheek statement.

Theorem 5. *Two choices suffice to lower the threshold for k-SAT for any k, assuming that the threshold conjecture holds.*

Proof. Our strategy depends on the topology of the formula, but in a very simple way. Let $0 < a < 1$ be a constant to be determined. We simply prefer clauses whose variables are all in the set $U = \{x_1, x_2, \ldots, x_{an}\}$ to those with one or more variables outside U.

If we have τ choices, the probability that the chosen clause has all its variables in U is

$$q = 1 - (1 - a^k)^\tau \,,$$

If the subformula consisting of these clauses is unsatisfiable, then so is the entire formula. But this subformula is uniformly random in $F_k(n', m')$ where $n' = an$ and $\mathbb{E}[m'] = qm$. By the Chernoff bound, its density is arbitrarily close to

$$\alpha' = \frac{m'}{n'} = \alpha\gamma \quad \text{where} \quad \gamma = \frac{1 - (1 - a^k)^t}{a} \,. \tag{12}$$

Thus the chosen formula is unsatisfiable w.h.p. if $\alpha > \alpha_k/\gamma$, lowering the threshold by a factor of γ.

To confirm that there is an a such that $\gamma > 1$, we maximize γ as a function of a. Specifically, if $\tau = 2$ then γ is maximized at

$$a = \left(\frac{2k-2}{2k-1}\right)^{1/k} \,,$$

where

$$\gamma = \frac{4k(k-1)}{(2k-1)^2}\left(\frac{2k-1}{2k-2}\right)^{1/k} \geq 1 + \frac{1}{4k^2} \,. \tag{13}$$

This completes the proof. □

We remark that a similar strategy shows that two choices are enough to create a giant component with $m = cn$ edges where $c = (3/8)\sqrt{3/2} = 0.459$.

What if we don't take the threshold conjecture for granted? Theorems 1–4 still "raise the threshold" unconditionally, in the sense that two or three choices give formulas that are w.h.p. satisfiable at densities where random k-SAT formulas are w.h.p. unsatisfiable. We can give an analogous result for lowering the threshold:

Theorem 6. *For any k, there is a constant τ such that τ choices suffice to generate formulas that are w.h.p. unsatisfiable at densities where random k-SAT formulas are w.h.p. satisfiable. For sufficiently large k, two choices suffice.*

Proof. Following the proof of Theorem 5, we just have to ensure that $\gamma > \gamma_k$ where $\gamma_k = \alpha_k^{\text{high}}/\alpha_k^{\text{low}}$ is the ratio between the best known upper and lower bounds on the threshold, i.e. the lowest and highest densities where random k-SAT formulas are known to be unsatisfiable or satisfiable respectively.

Examining (12), we see that for any k and any γ_k there are constants a, τ such that $\gamma > \gamma_k$. For instance, let $a = 1/(2\gamma_k)$ and let τ be large enough so that $(1 - a^k)^\tau < 1/2$.

For large k, from (2) we have $\gamma_k = 1 + O(2^{-k})$, where O represents a constant independent of k. Since from (13) we can achieve $\gamma = 1 + \Theta(1/k^2)$ with two choices, there is some k_0 such that two choices suffice for all $k \geq k_0$. □

For 3-SAT in particular, where the current value of γ_k is $4.898/3.52 = 1.275$, maximizing γ as a function of a shows that 6 choices suffice to lower the threshold unconditionally.

5 Conclusion

We have shown that three choices are enough to raise the satisfiability threshold in random k-SAT, and that two are enough to lower it, for any k. We have also shown that two are enough to raise it for $k \leq 50$. We are left with several open questions.

1. Are two choices enough to raise the threshold for any k? This seems incontrovertible, but we not see how to extend our analysis of Biased Unit Clause to arbitrary k.
2. Sinclair and Vilenchik [29] point out that if we are allowed to choose off-line, i.e. if we are given all pairs of clauses in advance, then with two choices can raise the k-SAT threshold exactly to the $2k$-SAT threshold, since a choice of two k-SAT clauses is equivalent to a $2k$-SAT clause. Can we do nearly this well in the on-line version? Or is there a stricter upper bound on how high we can raise the k-SAT threshold with two on-line choices, say $O(2^{ck})$ for some $c < 2$?
3. Our two-choice strategy for lowering the threshold does so by a factor of $1 + O(1/k^2)$. Is there a strategy with two choices, or a constant number of choices, that lowers the threshold by a constant factor for all k?

Acknowledgments. We are grateful to Stephan Mertens, Will Perkins, Danny Vilenchik, as well as the anonymous referees, for helpful conversations and suggestions for improvement.

References

1. Achlioptas, D., D'Souza, R., Spencer, J.: Explosive percolation in random networks. Science 323(5920), 1453–1455 (2009)
2. Achlioptas, D., Moore, C.: The asymptotic order of the random k-SAT threshold. In: Proc. 43rd Symposium on Foundations of Computer Science, pp. 779–788 (2002)
3. Achlioptas, D., Moore, C.: Almost all graphs with average degree 4 are 3-colorable. Journal Computer System Science 67(2), 441–471 (2003)
4. Achlioptas, D., Peres, Y.: The threshold for random k-SAT is $2^k(\ln 2 - O(k))$. In: Proc. 35th STOC, pp. 223–231 (2003)
5. Bohman, T., Frieze, A.: Avoiding a giant component. Random Struct. Algorithms 19(1), 75–85 (2001)
6. Bohman, T., Kravitz, D.: Creating a giant component. Combinatorics, Probability and Computing 15, 489–511 (2006)
7. Chao, M.-T., Franco, J.V.: Probabilistic analysis of two heuristics for the 3-satisfiability problem. SIAM Journal on Computing 15(4), 1106–1118 (1986)
8. Chao, M.-T., Franco, J.V.: Probabilistic analysis of a generalization of the unit clause literal selection heuristic for the k-satisfiability problem. Information Science 51, 289–314 (1990)
9. Chvatal, V., Reed, B.: Mick Gets Some (the Odds Are on His Side). In: Proc. 33rd FOCS, pp. 620–627 (1992)
10. Coja-Oghlan, A., Panagiotou, K.: Going after the k-SAT Threshold. To appear in Proc. STOC (2013)
11. Diaz, J., Kirousis, L., Mitsche, D., Perez, X.: On the satisfiability threshold of formulae with three literals per clause. Theoretical Computer Science 410, 2920–2934 (2009)
12. Dubois, O., Boufkhad, Y.: A general upper bound for the satisfiability threshold of random r-SAT formulae. Journal of Algorithms 24(2), 395–420 (1997)
13. Fernandez de la Vega, W.: Random 2-SAT: results and problems. Theoretical Computer Science 265(1-2), 131–146 (2001)
14. Flaxman, A., Gamarnik, D., Sorkin, G.: Embracing the giant component. Random Structures and Algorithms 27(3), 277–289 (2005)
15. Friedgut, E.: Sharp thresholds of graph properties, and the k-SAT problem. Journal of the American Mathematical Society 12(4), 1017–1054 (1999); Appendix by Bourgain, J.
16. Goerdt, A.: A Threshold for Unsatisfiability. Journal of Computer and System Sciences, 469–486 (1996)
17. Taghi Hajiaghayi, M., Sorkin, G.: The satisfiability threshold of random 3-SAT is at least 3.52. IBM Research Report RC22942 (2003)
18. Kalapala, V., Moore, C.: The Phase Transition in Exact Cover. Chicago Journal of Theoretical Computer Science 5 (2008)
19. Kaporis, A., Kirousis, L., Lalas, E.: The probabilistic analysis of a greedy satisfiability algorithm. Random Struct. Algorithms 28(4), 444–480 (2006)
20. Kirousis, L.M., Kranakis, E., Krizanc, D., Stamatiou, Y.C.: Approximating the unsatisfiability threshold of random formulas. Random Struct. Algorithms 12(3), 253–269 (1998)

21. Mertens, S., Mézard, M., Zecchina, R.: Threshold values of random K-SAT from the cavity method. Random Structures & Algorithms 28, 340–373 (2006)
22. Moore, C., Mertens, S.: The Nature of Computation. Oxford University Press (2011)
23. Mezard, M., Montanari, A.: Information, Physics and Computation. Oxford Graduate Texts (2009)
24. Mezard, M., Zecchina, R.: Random K-satisfiability problem: From an analytic solution to an efficient algorithm. Physical Review E 66, 056126, 1–26 (2002)
25. Mitchell, D., Selman, B., Levesque, H.: Hard and easy distributions of SAT problems. In: Proc. 10th. National Conference on Artificial Intelligence (AAAI), pp. 459–465 (1992)
26. Mossel, E., Sen, A.: Branching process approach for the 2-SAT threshold. Journal of Applied Probability 47(3), 796–810 (2010)
27. Perkins, W.: Random K-SAT and the power of two choices. arXIv:1209.5313v1 (September 2012)
28. Riordan, O., Wanke, L.: Achlioptas process phase transition are continuous. Annals of Applied Probability 22(4), 1450–1464 (2012)
29. Sinclair, A., Vilenchik, D.: Delaying Satisfiability for Random 2SAT. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, pp. 710–723. Springer, Heidelberg (2010)
30. Spencer, J., Wormald, N.: Birth Control for Giants. Combinatorica 27, 587 (2007)
31. Verhoeven, N.C.: Random 2-SAT and unsatisfiability. Information Professing Letters 72(3-4), 119–124 (2000)
32. Wormald, N.C.: Differential equations for random processes and random graphs. Annals of Applied Probability 5, 1217–1235 (1995)

Connectivity of Random High Dimensional Geometric Graphs

Roee David and Uriel Feige

Department of Computer Science and Applied Mathematics,
The Weizmann Institute of Science, Rehovot, Israel
{roee.david,uriel.feige}@weizmann.ac.il

Abstract. We consider graphs obtained by placing n points at random on a unit sphere in $\mathbb{R}^d$, and connecting two points by an edge if they are close to each other (e.g., the angle at the origin that their corresponding unit vectors make is at most $\pi/3$). We refer to these graphs as *geometric graphs*. We also consider a complement family of graphs in which two points are connected by an edge if they are far away from each other (e.g., the angle is at least $2\pi/3$). We refer to these graphs as *anti-geometric graphs*. The families of graphs that we consider come up naturally in the context of semidefinite relaxations of graph optimization problems such as graph coloring.

For both distributions, we show that the largest dimension for which a random graph is likely to be connected is the same (up to an additive constant) as the largest dimension for which a random graph is likely not to have isolated vertices. The phenomenon that connectivity of random graphs is tightly related to nonexistence of isolated vertices is not new, and appeared in earlier work both on nongeometric models and on other geometric models. The fact that in our model the dimension d is allowed to grow as a function of n distinguishes our results from earlier results on connectivity of random geometric graphs.

1 Introduction

For natural numbers n and d and an angle $0 < \theta < 2\pi$, an (n, θ, d)-graph is a collection of n points on the unit sphere in $\mathbb{R}^d$ (equivalently, n unit vectors), with two points (vertices) connected by an edge iff the angle between their corresponding unit vectors is at most θ. For example, when $d = 2$, we have points placed on a circle of radius 1, and two points are neighbors if the angular distance between them is at most θ. Equivalently, each point can be viewed as representing an interval of angular length θ centered at the point, and two intervals are neighbors if they intersect. When $d \geq 3$ the points can be thought of as equal size discs placed on the unit sphere, and two points are neighbors if their discs intersect. In this work we shall be interested in the case when the dimension d is relatively large and scales roughly like $\log n$. We call such graphs high dimensional geometric graphs. Our interest in high dimensional geometric graphs (and their complements that we call *anti-geometric* graphs,

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 497–512, 2013.

with an edge iff the angle is at least θ) stems from the fact that they come up naturally as solutions for various semidefinite relaxations of combinatorial optimization problems (see [13,10], for example). The question that we address in the current work is that of connectivity of these graphs. Specifically, we consider the distribution $G_{n,d,\theta}$ in which the locations on the sphere of the n vertices of the (n,θ,d)-graph are chosen uniformly independently at random, and ask for which range of parameters of (n,θ,d) is the graph likely to be connected. This question comes up naturally in the study of algorithms for coloring of random high dimensional anti-geometric graphs [4], and turns out to be more subtle than one might first imagine.

1.1 Definitions and Notation

We use $\mathbb{S}^d$ to denote the unit sphere centered at the origin in $\mathbb{R}^d$, namely, the set of vectors in $\mathbb{R}^d$ of Euclidean norm 1. (Note that in other literature this is sometimes denoted by $\mathbb{S}^{d-1}$, due to the fact that it is a $(d-1)$-dimensional object.) We measure the distance between two points in $\mathbb{S}^d$ by the angle at the origin between the unit vectors that represent these points. Namely, for unit vectors u and v, their *angular distance* is $\arccos(vu)$, where uv denotes their inner product. An (n,θ,d)-graph $G(V,E)$ has as its vertex set V a collection of n points in $\mathbb{S}^d$, and for $u,v \in V$ there is an edge $(u,v) \in E$ iff their angular distance is at most θ, namely, $\arccos(vu) \leq \theta$. We refer to these graphs as *geometric graphs*, and to θ as the *neighborhood radius*. Observe that as we measure angular distance, a geometric graph remains unchanged if we scale the radius of $\mathbb{S}^d$ to be different than 1. We shall also be interested in complements of (n,θ,d)-graphs, in which $(u,v) \in E$ iff $\arccos(vu) \geq \theta$. We refer to these graphs as *anti-geometric graphs*, and to $\pi - \theta$ as the neighborhood radius. (We note that once we fix θ, in our graphs it will not happen that $\arccos(vu) = \theta$, and hence geometric and anti-geometric graphs are indeed complements of each other.)

Related work studied other families of geometric graphs in $\mathbb{R}^d$ (not necessarily on $\mathbb{S}^d$), with geometric distances induced either by the Euclidean norm or by other norms. We shall use the term *neighborhood radius*, typically denoted by r, to denote the geometric distance up to which two points are declared to be neighbors in these models as well.

For given (n,θ,d), we shall be interested in the distribution $G_{n,\theta,d}$ over (n,θ,d)-graphs, in which the n vertices are placed independently uniformly at random in $\mathbb{S}^d$. Likewise, $\bar{G}_{n,\theta,d}$ denotes the distribution over anti-geometric graphs (the complements of (n,θ,d)-graphs) when the n vertices are placed independently uniformly at random in $\mathbb{S}^d$.

Given a graph $G(V,E)$ and two (not necessarily disjoint) sets A and B of vertices, $E_G(A,B)$ denotes the set of edges with one endpoint in A and the other endpoint in B, and $e_G(A,B) = |E_G(A,B)|$ denotes the number of such edges.

Definition 1. *A graph $G(V,E)$ is called an* (n,h)-expander *if $|V| = n$ and for every set of vertices $S \subset V$ with $|S| \leq n/2$ it holds that $e(S, V \setminus S) \geq h|S|$.*

1.2 Our Results

We shall assume that θ is a fixed constant (specifically, $\theta = \pi/3$), and consider increasing values of d.

Given d, scale the sphere $\mathbb{S}^d$ so that its total surface area is 1. Let $\mu(\theta, d)$ denote the area of a spherical cap of angular radius θ. Given a vertex in $\mathbb{S}^d$, its expected number of neighbors in $G_{n,\theta,d}$ is precisely $(n-1)\mu(\theta, d)$. Hence if $n < 1/\mu(\theta, d)$ we expect a constant fraction of the vertices to be isolated (implying that the graph is not connected). It is not difficult to show (see Section 5) that a value of $n_{IV} \simeq \frac{1}{\mu(\theta,d)} \ln(1/\mu(\theta, d))$ serves as a threshold value for isolated vertices: for every $\epsilon > 0$, if $n \geq (1+\epsilon)n_{IV}$ there are unlikely to be isolated vertices, whereas if $n \leq (1-\epsilon)n_{IV}$ there are likely to be isolated vertices. The same applies to $\bar{G}_{n,\theta,d}$ by setting $n_{IV} \simeq \frac{1}{\mu(\pi-\theta,d)} \ln(1/\mu(\pi-\theta, d))$.

Theorem 1. *Let $\theta < \pi/2$ be a fixed constant (e.g., $\theta = \pi/3$) and let d be sufficiently large. There is some universal constant $c \geq 1$ such that if $n \geq cn_{IV}$ then both $G_{n,\theta,d}$ and $\bar{G}_{n,\pi-\theta,d}$ are connected with probability $1 - o(1)$ (where the $o(1)$ term tends to 0 as d grows).*

Our proof of Theorem 1 shows that a value of $c \simeq \pi/\theta$ suffices. One may conjecture that Theorem 1 is true also for some absolute constant c independent of θ, and moreover, that this constant is not much larger than 1.

We are mostly interested in the case that θ is constant, d is a parameter that may grow, and n is exponential in d. Given θ and d, Theorem 1 determines up to a constant multiplicative factor the smallest value of n for which the graph is likely to be connected. Had we fixed θ and n, the same proof would determine up to a constant *additive* term the largest value of d for which the graph is likely to be connected. For example, if $\theta = \pi/3$ and one is given a value of n, our results establish that there is a constant $c_1 \simeq 6.95212$ (see Section 5) and a constant $c_2 > 0$ such that if $d \leq c_1 \log n$ the graph is connected with probability $1 - o(1)$, and if $d \geq c_1 \log n + c_2$ the graph is connected with probability at most $o(1)$. Finally, had we fixed d and n (exponential in d), the proof of Theorem 1 would determine up to an additive term of order $O(1/d)$ the smallest value of θ for which the graph is likely to be connected.

Our proof of Theorem 1 involves two aspects. One is that of establishing various expansion properties of a random (N, θ, d) graph G' when θ is a fixed constant (e.g., $\theta = \pi/3$), d is a parameter that can grow, and N tends to infinity. Establishing these properties involves symmetrization arguments of the type used in [6,7]. Thereafter, we view the random (n, θ, d) graph G as a random subgraph of G' induced on n random vertices. We wish to show that the expansion properties of G' imply that G is likely to be connected. This is done using the following theorem which is applied in a situation in which the expansion h is of the order of the maximum degree in G, and this maximum degree is so large (e.g., it might be $\frac{N}{\log N}$) so that $\log(N/h)$ is much smaller than $\log N$.

Theorem 2. *Let $G(V, E)$ be an (N, h)-expander of maximum degree Δ. Consider a vertex induced subgraph H that contains qN vertices chosen randomly*

and independently. If $q > \frac{1+o(1)}{h}\ln(\frac{N}{h}) + O\left(\frac{1}{h}\log(\frac{\Delta}{h})\right)$ *then the probability that* H *is connected is at least* $1 - o(1)$*, where the* $o(1)$ *term tends to 0 as* $\frac{N}{h}$ *tends to infinity.*

1.3 Related Work

Connectivity in uniform random graphs (non-geometric) is well understood. Given n vertices which initially have no edges, if one inserts random and independent edges to the graph, with high probability the graph becomes connected exactly at the point where every vertex has degree at least one [5]. Theorem 1 is an approximate version of this tight connection between connectivity and having no isolated vertices. Theorem 1 has several precursors in work on connectivity of various models of geometric graphs [16]. In the most general setting, a "nice" (in particular, connected) domain $D \in \mathbb{R}^d$ is given together with a measure on the domain and a norm. (In our setting the domain is the unit sphere $\mathbb{S}^d$, the measure is uniform, and the norm can be taken to be Euclidean.) One places n points at random in D, and two points are neighbors if the distance between them (according to the given norm) is at most r. The question typically asked is what is the minimum value of r (as a function of n) for which the graph is likely to be connected. We denote the expectation of this value by $R_C(n)$. Equivalently, given the n points one considers a minimum spanning tree (with edge lengths being the distance between the points according to the given norm), and asks what is the length of the longest edge in this spanning tree. It is not hard to see that this is exactly the value $R_C(n)$ that would ensure connectivity. A different question is what is the minimum value of r for which there are not likely to be any isolated vertices. We call the expectation of this value $R_{IV}(n)$. Clearly, $R_C(n) \geq R_{IV}(n)$ for every n. A very general result of Penrose [19] shows that as n tends to infinity, the ratio $\frac{R_C(n)}{R_{IV}(n)}$ tends to 1. There were previous results of this nature in special cases. See [8,2,12,18], for example.

It might appear that the result of Penrose (or other related previous work) implies our Theorem 1 as a special case. However, this overlooks the issue that in previous results that we are aware of, one first fixes the dimension d and then lets n tend to infinity (we refer to this as *asymptotic* n), whereas in our results n is fixed as a function of d (we refer to this as *bounded* n). As a consequence, the statement of previous results only implies that if θ is sufficiently small as a function of d, a theorem such as Theorem 1 holds. In contrast, we prove Theorem 1 when θ is a fixed constant. Once θ is sufficiently small the corresponding graphs acquire geometric properties that are different than those involved in the case when θ is large (when θ is small graph distances approximate well geometric distances on the sphere, whereas when θ is large this is no longer true), and hence proofs of connectivity that apply in one setting might not apply in the other. This aspect is discussed further in the full version of our paper.

Another line of work related to the questions studied in this work is that of *percolation* (see [11] or [17], for example), and specifically, the process referred to as *continuum percolation*. Typically, in the continuum percolation process

one is given a domain D such as a box in $\mathbb{R}^d$ that is symmetric around the origin. One places a point in the origin and additional points at random in the domain, and two points are connected if their distance from each is at most r. The question asked is not that of complete connectivity of the graph, but rather questions such as whether there is a path from the origin to the boundary of the domain, or what is the size of the connected component that contains the origin. A typical situation is that for large enough n there is a given threshold distance R_p such that if $r > R_p$ the origin is likely to be connected to the boundary, whereas if $r < R_p$ the origin is likely to be in a very small component, or even isolated. Obviously, the connectivity threshold is at least as large as the percolation threshold, namely $R_C \geq R_p$. Some of the work on connectivity builds on results from percolation (e.g.,[12] explicitly refers to results in [15], and [19] makes use of proof techniques, a Peierls argument, that is commonly used in percolation). We remark that in percolation theory one typically deals with the regime of asymptotic n rather than bounded n.

The results of [9] may serve to illustrate a difference between our setting of fixed n and the setting that n tends to infinity. Their result is that every monotone graph property (any property that is preserved by adding edges, connectivity being one such example) in random geometric graphs (in their case, the domain is $[0,1]^d$ and the measure is uniform) has a sharp threshold. More specifically, for every n and for every monotone graph property, there is a corresponding threshold distance R (that depends on n and on the property), such that if $r > R+\epsilon$ the property holds almost surely, and if $r < R-\epsilon$, the property almost surely does not hold. Moreover, the value of ϵ tends to 0 as n grows, specifically at a rate $\epsilon \leq O((\frac{\log n}{n})^{1/d})$. In our case of constant θ, the values of n that we consider are only exponential in d, and consequently the results only imply that $\epsilon \leq O(1)$. Moreover, if one inspects the proof technique of [9], it necessarily results in $\epsilon = \Omega(1)$ when θ is constant. As we consider the unit sphere rather than unit cube, this value of ϵ is of the same order of magnitude as the diameter of the whole domain, and hence completely useless. (We do not claim that monotone graph properties do not have sharp thresholds when $n \leq 2^{O(d)}$. We just point out that if they do, establishing this will require proofs that are different than those that apply when n tends to infinity.)

Theorem 2 considers connectivity of random vertex induced subgraphs of expanders. There have been previous studies of connectivity properties of random subgraphs of expanders. However, all previous work that we are aware of either addressed edge induced subgraphs (see for example [1]), or addressed vertex induced subgraphs at a range of parameters that is very different from that of Theorem 2 (see for example [3]).

Low dimensional geometric graphs are sometimes used as models for wireless communication networks (e.g., in [12]), or for physical medium (see for example Chapter 1 in [11]). Random high dimensional geometric graphs such as the ones studied in this paper are not commonly used as a model for physical reality. However, high dimensional geometric graphs come up naturally as solutions to semidefinite programming relaxation to various optimization problems. In particular, the

solution to semidefinite relaxations of graph coloring problems are high dimensional anti-geometric graphs (see [13], though note that the terminology anti-geometric graphs is not used there). Random *anti-geometric* graphs are used in order to construct negative examples showing a large (tight, in some cases) integrality gap for these semidefinite relaxations [6,7]. The issue of connectivity of these graphs did not come up in these earlier works, but did come up and was left open in more recent study of these negative examples [4].

2 On the Expansion of the Infinite Graph

In this section we define an infinite graph $\mathbb{G}_{\theta,d}$, with the property that an induced graph on a random sample of n vertices from $\mathbb{G}_{\theta,d}$ is distributed as $G \in_R G_{n,\theta,d}$. We will show (under a natural definition of expansion of infinite graphs) that $\mathbb{G}_{\theta,d}$ is an expander.

Given $\boldsymbol{v} \in \mathbb{S}^d$ the set $\left\{\boldsymbol{u} \in \mathbb{S}^d \middle| \arccos(\boldsymbol{v}\boldsymbol{u}) \leq \theta\right\}$ is a *sphere cap* and it is denoted by $C_{\cos(\theta)}(\boldsymbol{v})$. For example, a cap of angular radius $\pi/3$ centered at $\boldsymbol{v}$ is denoted by $C_{\frac{1}{2}}(\boldsymbol{v})$ and a cap of radius $\pi/2$ (a half sphere) is denoted by $C_0(\boldsymbol{v})$. We omit $\boldsymbol{v}$ from the above notation if the location of the center of the cap is not needed. (The subscript denotes $\cos\theta$ rather than simply θ for compatibility with notation in previous work.)

Definition 2. *(The infinite graph $\mathbb{G}_{\theta,d}$). The vertices of $\mathbb{G}_{\theta,d}$ are all the points in $\mathbb{S}^d$ and the edge set of $\mathbb{G}_{\theta,d}$ is all the pairs $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{S}^d$ s.t. $\boldsymbol{u} \in C_{\cos\theta}(\boldsymbol{v})$.*

Every vertex in $\mathbb{G}_{\theta,d}$ has an infinite number of neighbors, therefore the notion of the "degree" of a vertex is represented by the measure of the set of its neighbors. We normalize the uniform measure on the sphere so that the total measure of the (surface area of the) sphere is 1. Given a measurable set S, its measure (the ratio between the surface area of S to the surface area of $\mathbb{S}^d$) is denoted by $|S|$. This measure corresponds to the number of vertices in S. The edge boundary of S (the set of edges with exactly one endpoint in S) is denoted by $\partial(S)$ and its measure is denoted by $|\partial(S)|$. This measure corresponds to the number of edges leaving S. (The measure of $\partial(S)$ is the integral over all points in S of the measures of the sets of their neighbors outside S.)

The following definition for edge expansion is given, for simplicity, specificity for $\mathbb{G}_{\theta,d}$. An equivalent definition to a general infinite graph can be stated in a straightforward manner.

Definition 3. *(Edge expansion for the infinite graph). The edge expansion $h(\mathbb{G}_{\theta,d})$ of a graph $\mathbb{G}_{\theta,d}$ is defined as*

$$h(G) = \inf_{0<|S|\leq\frac{1}{2}} \frac{|\partial(S)|}{|S|}$$

where the minimum is over all measurable sets $S \subset \mathbb{S}^d$ with nonzero measure. For a given set S we call the quantity $h(G,S) := \frac{|\partial(S)|}{|S|}$ the edge expansion of S.

An expander graph is a graph with high edge expansion. To lower bound $h\left(\mathbb{G}_{\theta,d}\right)$ it suffices to show that the sets with the lowest edge expansion are sphere caps (Lemma 1) and to analyze the expansion of spherical caps (Lemma 2).

Lemma 1. *For all $S \subset \mathbb{S}^d$ s.t. $|S| = m$ $(0 \leq m \leq \frac{1}{2})$ it holds that $|\partial(S)| \geq |\partial(C_a)|$ for the unique a satisfying $|C_a| = m$.*

Proof. The proof is a direct consequence of Theorem 5 in [7]. The theorem shows that the complement of the graph $\mathbb{G}_{\theta,d}$ (u,v are neighbors in the complement graph iff they are not neighbors in the original graph) has the following property: for all $S \subset \mathbb{S}^d$ s.t. $|S| = m$ $(0 \leq m \leq \frac{1}{2})$ it holds that $|\partial(S)| \leq |\partial(C_a)|$ for the unique a satisfying $|C_a| = m$ (here $\partial(S)$ is the edge boundary of the complement graph of $\mathbb{G}_{\theta,d}$). □

In the rest of this section we deal with $\theta = \pi/3$ for geometric graphs and $\theta = 2\pi/3$ for *anti-geometric* graphs, though the results generalize for any θ.

Lemma 2. *$h\left(\mathbb{G}_{\pi/3,d}\right) \geq (1/3 - \epsilon)\left|C_{\frac{1}{2}}\right|$ for ϵ that tends to 0 as d grows.*

Proof. By Lemma 1, the minimum expansion of $\mathbb{G}_{\pi/3,d}$ is attained at a sphere cap. It is not difficult to see that the expansion of sphere caps decreases as their radius increases. Hence among sets of measure at most 1/2, the minimum expansion is attained for the half sphere C_0. Hence $h\left(\mathbb{G}_{\pi/3,d}\right) = h\left(\mathbb{G}_{\pi/3,d}, C_0\right)$. Estimating $h\left(\mathbb{G}_{\pi/3,d}, C_0\right)$ is fairly simple once d is sufficiently large, as we show below.

Fix $\epsilon' > 0$ to be a small constant. For every vertex $\boldsymbol{v}$ of $\mathbb{G}_{\pi/3,d}$ remove those edges to neighbors of $\boldsymbol{v}$ with angle smaller than $\pi/3 - \epsilon'$, thus obtaining a new graph $\mathbb{G}'_{\pi/3,d}$. The ratio $\frac{C_{\cos(\pi/3-\epsilon')}}{C_{\cos(\pi/3)}}$ tends to zero as d grows. (See Theorem 6.) This implies that we have removed only a small fraction (that tends to zero) of the graph edges.

Consider $\boldsymbol{y}$ drawn uniformly from $\mathbb{S}^d$. Any edge e of $\mathbb{G}'_{\pi/3,d}$ has probability $\frac{\pi/3-\epsilon'}{\pi} = 1/3 - \epsilon'/\pi$ to be in $\partial\left(C_0\left(\boldsymbol{y}\right)\right)$ (the analysis is similar to that of the random hyperplane rounding technique of [10]). Hence roughly one third of the edges of the graph are in the edge boundary of C_0. As for the remaining edges, by symmetry half of them are in the half sphere C_0 and half are in its complement. Hence there are essentially as many edges in the edge boundary of C_0 as there are inside C_0, implying that the expansion of C_0 is nearly one third of (the measure of) the degree of its vertices, establishing that that $h\left(\mathbb{G}_{\pi/3,d}, C_0\right) = (1/3 - \epsilon)\left|C_{\frac{1}{2}}\right|$, as desired.

□

2.1 The Expansion of the Infinite Anti-geometric Graph

The infinite anti-geometric graph $\bar{\mathbb{G}}_{\pi/3,d}$ is defined as follows: its vertices are all the points in $\mathbb{S}^d$ and the edge set of $\bar{\mathbb{G}}_{\pi/3,d}$ is all the pairs $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{S}^d$ s.t. $\boldsymbol{u} \in C_{\frac{1}{2}}\left(-\boldsymbol{v}\right)$, (as opposed to $\boldsymbol{u} \in C_{\frac{1}{2}}\left(\boldsymbol{v}\right)$ in the case of $\mathbb{G}_{\pi/3,d}$). The graph induced on a random sample of n vertices from $\bar{\mathbb{G}}_{\pi/3,d}$ is distributed as $G \in_R \bar{G}_{n,\pi/3,d}$. The edge expansion of $\bar{\mathbb{G}}_{\pi/3,d}$, $h\left(\bar{\mathbb{G}}_{\pi/3,d}\right)$, is defined similarly as in Definition 3.

Lemma 3. $h\left(\bar{\mathbb{G}}_{\pi/3,d}\right) \geq h\left(\mathbb{G}_{\pi/3,d}\right)$

Proof. In this proof we shall switch between several graphs. Given a graph H and sets A, B of vertices, the set edges of H with one endpoint in A and the other in B will be denoted by $E\left(H, A, B\right)$. Given $A \subset \mathbb{S}^d$ let $\bar{A} := \mathbb{S}^d \setminus A$, i.e the complement set, and let $A^{-1} := \left\{\boldsymbol{x} \in \mathbb{S}^d \middle| -\boldsymbol{x} \in A\right\}$.

To prove the lemma we need to show that

$$\min_{|A|=a} \left|E\left(\bar{\mathbb{G}}_{\pi/3,d}, A, \bar{A}\right)\right| \geq \min_{|A|=a} \left|E\left(\mathbb{G}_{\pi/3,d}, A, \bar{A}\right)\right|$$

for every $0 < a \leq \frac{1}{2}$. Consider an arbitrary $0 < a \leq \frac{1}{2}$.

$$\min_{|A|=a} \left|E\left(\bar{\mathbb{G}}_{\pi/3,d}, A, \bar{A}\right)\right| \geq \min_{\substack{|A|=a \\ |B|=1-a}} \left|E\left(\bar{\mathbb{G}}_{\pi/3,d}, A, B\right)\right|$$

$$= \min_{\substack{|A|=a \\ |B|=1-a}} \left|E\left(\mathbb{G}_{\pi/3,d}, A, B^{-1}\right)\right| = \min_{\substack{|A|=a \\ |B|=1-a}} \left|E\left(\mathbb{G}_{\pi/3,d}, A, B\right)\right|$$

To finish the proof it suffices to show that:

$$\min_{\substack{|A|=a \\ |B|=1-a}} \left|E\left(\mathbb{G}_{\pi/3,d}, A, B\right)\right| = \min_{|A|=a} \left|E\left(\mathbb{G}_{\pi/3,d}, A, \bar{A}\right)\right|$$

We claim that this last equality is a consequence of Theorem 3.5 in [6], which shows the following:

Consider the infinite anti-geometric graph defined on $\mathbb{S}^d$ with parameter θ (in our case we shall take $\theta = \pi - \pi/3$). Let $0 < a \leq 1$ and let A and B be two (not necessarily disjoint) measurable sets in $\mathbb{S}^d$ of measure a. Let $\boldsymbol{x}$ be an arbitrary vertex of $\mathbb{S}^d$. The minimum of $|E(A, B)|$ is obtained when $A = B = C_b\left(\boldsymbol{x}\right)$ where C_b is a cap of measure $|C_b| = a$.

By the above theorem $A = B = C_b\left(\boldsymbol{x}\right)$ minimizes

$$\min_{\substack{|A|=a \\ |B|=a}} \left|E\left(\bar{\mathbb{G}}_{\pi-\pi/3,d}, A, B\right)\right|.$$

Since $\mathbb{G}_{\pi/3,d}$ and $\bar{\mathbb{G}}_{\pi-\pi/3,d}$ are complement graphs of each other, $A = B = C_b\left(\boldsymbol{x}\right)$ maximizes

$$\max_{\substack{|A|=a \\ |B|=a}} \left|E\left(\mathbb{G}_{\pi/3,d}, A, B\right)\right|$$

Equivalently, $A = C_b\left(\boldsymbol{x}\right)$ and $B = \bar{A}$ maximize

$$\max_{\substack{|A|=a \\ |B|=1-a}} \left|E\left(\mathbb{G}_{\pi/3,d}, A, \bar{B}\right)\right|.$$

By regularity of the graph $\mathbb{G}_{\pi/3,d}$, it follows that $A = C_b(\boldsymbol{x})$ and $B = \bar{A}$ minimize

$$\min_{\substack{|A|=a \\ |B|=1-a}} \left|E\left(\mathbb{G}_{\pi/3,d}, A, B\right)\right|$$

proving the claim. □

3 Connectivity of Random Geometric and Anti-geometric Graphs

Having established expansion properties for the infinite graph, we present two proofs of Theorem 1. One proof first "discretizes" the infinite graph, thus obtaining a nearly regular very dense finite graph with expansion properties similar to that of the infinite graph (expansion roughly one third of the degree, for our choice of $\theta = \pi/3$). This dense graph can be thought of as being obtained by taking a finite though extremely large number N of sample points from $\mathbb{G}_{\pi/3,d}$. The formal details of such a discretization are similar to those in [7,6], and are omitted here. Thereafter, noting the relation $h \geq 0.3\Delta$ between the expansion and maximum degree, one can use Theorem 2, whose proof appears in Section 4.

The other way to prove the main theorem is via a direct proof of sampling from the infinite graph, without performing the discretization first. This may appear in the full version of the paper.

We note that there are alternative approaches that can be used in order to try to prove connectivity of geometric graphs. Specifically, one may try to establish that the graph enjoys a property called *geometric routing*. Essentially, this property means that between every two vertices u and v of the graph there is a path that respects the geometry of the sphere – advancing from u to v along this path decreases the geometric distance to v in every step. However, for our graphs, geometric routing will not work. In fact, the number n of vertices that are required in order to have geometric routing in $G \in_R G_{n,\pi/3,d}$ is such that the average degree of the graph is as high as roughly $n^{0.29}$, rather than only $O(\log n)$ which (as Theorem 1 shows) suffices for connectivity. See more details in the full version of this paper.

4 Connectivity of Subgraphs of Expanders

In this section we prove Theorem 2. We were tempted to try some simpler proof techniques than the ones used in this section, but encountered difficulties in employing them. This issue is discussed further in the full version of this paper.

We shall be concerned with a graph G that is an (n, h)-expander of maximum degree Δ, and a random vertex induced subgraph of G that we denote by H. We use the notation $V, E, n, N(S)$ for set of vertices, set of edges, number of vertices, set of neighbors of S not including S, respectively, all in relation to the graph G.

Our proof of Theorem 2 involves two steps.

1. The first step is similar in nature to percolation. We pick an arbitrarily sampled vertex $v \in V_H$ and then show that with high probability it belongs to a fairly large connected component CC_v in H. The size of the connected component is not measured in terms of the number of vertices that it contains, but rather in terms of the fraction of vertices of the original expander graph G that are neighbors of this connected component. We show that this fraction to be at least half. Namely, $|N_G(CC_v) \bigcup CC_v| \geq n_G/2$.
2. The second step shows that with high probability every vertex $u \in V_H$ has a path consisting only of vertices from H that connects it to CC_v. This step uses the fact that $N_G(CC_v)$ is large, and hence is easy to reach.

In our analysis of Step 1 we shall use the following lemma.

Lemma 4. *For arbitrary $0 < \mu \leq \Delta \leq M$, let $x_i, x_2, \ldots$ be a sequence of nonnegative random variables satisfying $x_i \leq \Delta$ for all i, $E[x_1] \geq \mu$, and $E[x_i|x_1, \ldots, x_{i-1}] \geq \mu$ for $i \geq 2$. Let t be a stopping time, giving the smallest index such that $\sum_{i=1}^{t} x_i \geq M$. Then the following hold:*

1. $E[t] \leq \frac{M+\Delta}{\mu}$.
2. $Pr[t > 2\frac{M}{\mu}] \leq \frac{2\Delta}{M}$.

Proof. Change every random variable x_i to a nonnegative random variable x'_i of expectation exactly μ by reducing its value, if needed. Consider the sequence $y_1, y_2, \ldots$ of random variables in which $y_i = x'_i - \mu$ for all i. The sequence $Y_i = \sum_{j=1}^{i} y_j$ is a Martingale. Let t be a stopping time for the Martingale sequence, giving the smallest index such that $Y_t \geq M - t\mu$. The random variable t has bounded moments, and hence by the optional stopping time theorem for martingales, $E[Y_t] = 0$. Moreover, by the fact that $y_i \leq \Delta - \mu$ and the minimality of t, we have $Y_t \leq Y_{t-1} + (\Delta - \mu) < M - (t-1)\mu + \Delta - \mu = M + \Delta - t\mu$. Hence $E[M + \Delta - t\mu] > 0$ implying $E[t] \leq \frac{M+\Delta}{\mu}$, proving item 1 of the lemma.

Let σ_i^2 be the variance of y_i (conditioned on $y_1, \ldots, y_{i-1}$). Observe that $\sigma_i^2 \leq \frac{\mu}{\Delta}(\Delta - \mu)^2 + \frac{\Delta - \mu}{\Delta}\mu^2 \leq \mu\Delta$. Using the fact that $E[y_i|y_j] = 0$ for $i > j$ we obtain that $var(Y_t) \leq \mu\Delta t$, implying by Chebychev's inequality that $Pr[Y_t < -\mu t/2] \leq \frac{4\Delta}{\mu t}$. For $t = 2\frac{M}{\mu}$ this gives $Pr[Y_t < -M] \leq \frac{2\Delta}{M}$. Observe that $X_{2M/\mu} \geq Y_{2M/\mu} + 2M$, implying item 2 of the lemma. □

The bounds we shall use for Step 1 will be presented in Theorem 4. We first prove Theorem 3 which presents bounds that are incomparable to those of Theorem 4, and whose proof can serve as an introduction to the proof technique of Theorem 4.

Theorem 3. *Let $G(V,E)$ be an n-vertex Δ-regular graph with edge expansion at least h, and let $r \in V$ be an arbitrary vertex. Then with probability at least $1/2$, a random sample U of $\frac{4n}{h} \ln \frac{n}{\Delta}$ vertices contains a subset $S \subset U \cup \{r\}$ with $|N(S) \cup S| \geq n/2$ such that the subgraph induced on S is connected.*

Proof. We expose vertices of U one by one. For $1 \leq i \leq |U|$, let u_i be the ith vertex exposed and let $U_i = \{u_1, \ldots u_i\}$. At every step i of the process we shall maintain a subset $S_i \subset U_i \cup \{r\}$ with $r \in S_i$ such that the subgraph induced on S_i is connected. Specifically, $S_0 = \{r\}$, and $u_i \in S_i$ iff $u_i \in N(S_{i-1})$.

Let us track the growth of $|S_i \cup N(S_i)|$. Initially, $|S_0 \cup N(S_0)| = 1 + \Delta$. For $i \geq 1$ we have that $|S_i \cup N(S_i)| \leq |S_{i-1} \cup N(S_{i-1})| + \Delta$. By the expansion properties of G and averaging arguments, the expected growth in step i satisfies $E[|(S_i \cup N(S_i)) \setminus (S_{i-1} \cup N(S_{i-1}))|] \geq \frac{|S_{i-1} \cup N(S_{i-1})|}{n} h$, as long as $|S \cup N(S)| \leq n/2$.

Partition the growth of S_i into phases, where phase ℓ ends at the smallest value of i for which $|S_i \cup N(S_i)| \geq \ell\Delta$. Then by item 1 of Lemma 4 the expected number of steps that phase $\ell + 1$ takes is at most $\frac{2\Delta n}{h\ell\Delta} = \frac{2n}{h\ell}$. It takes $\frac{n}{2\Delta}$ phases to reach $|S \cup N(S)| \geq n/2$. Hence the expected number of steps required is at most $\sum_{\ell=2}^{\frac{n}{2\Delta}} \frac{2n}{h\ell} < \frac{2n}{h} \ln \frac{n}{\Delta}$. The Theorem follows from Markov's inequality. □

In the statement of Theorem 4 and in its proof, c denotes some sufficiently large constant independent of n, Δ, h.

Theorem 4. *Let $G(V,E)$ be an n-vertex Δ-regular graph with edge expansion at least h, and let $r \in V$ be an arbitrary vertex. Then with probability at least $1/2$, a random sample U of $\frac{cn}{h} \log \frac{\Delta}{h}$ vertices contains a subset $S \subset U \cup \{r\}$ with $|N(S) \cup S| \geq n/2$ such that the subgraph induced on S is connected.*

Proof. Partition U into two parts U' and $U"$ of equal size (namely, $|U'| = |U"| = |U|/2$). A proof similar to that of Theorem 3 (details omitted) implies that with overwhelming probability, U' suffices in order to grow S from its initial size of $|S_0 \cup N(S_0)| = 1 + \Delta$ by a factor of 8, reaching size $|S \cup N(S)| = 8\Delta$. It remains to show that $U"$ can be used in order to grow S further, eventually reaching $|S \cup N(S)| = n/2$.

We expose information about vertices of $U"$ only when needed. The exposure algorithm will proceed in phases. Let $k = \log(4\Delta/h)$. Initially $U"$ is partitioned into $k+1$ sets, $U_0^0, U_0^1, \ldots U_0^k$. Renaming S_0 to be the outcome of the first part, we have $|S_0 \cup N(S_0)| \geq 8\Delta$. We shall have $|U_0^0| = \frac{2n}{h}$, $|U_j^1| = \frac{8n}{h}$ for every $1 \leq j \leq k$. Each phase i is composed of k subphases, where in the jth subphase one scans U_{i-1}^j. In such a scan some vertices are moved to S_{i-1} thus eventually obtaining S_i. The vertices moved to S are replaced in the respective U_{i-1}^j by fresh vertices from U_{i-1}^0. Hence the cardinalities of U^j for $1 \leq j \leq k$ remain unchanged during the exposure algorithm, and $|U^0|$ decreases exactly at the same rate by which $|S|$ increases.

A subphase is considered *successful* if the size of $S \cup N(S)$ increases by a multiplicative factor of at least 2 during the subphase.

Given the current S, a vertex u that is scanned is *good* if $u \in N(S)$ and moreover, $|N(S \cup \{u\})| \geq |N(S)| + \frac{h}{4} + 1$. Only good vertices are added to

S. Observe that the total number of good vertices cannot exceed $\frac{n/2}{h/4}$ (while maintaining $|S \cup N(S)| \leq n/2$), and hence U^0 can indeed compensate for all good vertices.

We now compute the expected contribution of a scanned vertex u, conditioned on all previous subphases being successful. Let S' be the set S at the time u was scanned in the previous phase. Unless u is a fresh vertex from U^0, we need to condition on $u \notin S' \cup N(S')$. Because the previous $k-1$ subphases were successful we have $|N(S) \cup S| \geq \frac{2\Delta}{h}|N(S') \cup S'|$. By the expansion properties of G, the number of edges exiting $N(S) \cup S$ is at least $h(N(S) \cup S)$. At least $\frac{3h}{4}(N(S) \cup S)$ of these edges originate from vertices of $N(S)$ that have at least $h/4$ exiting edges. At most $\Delta\frac{h}{2\Delta}(N(S) \cup S) = \frac{h}{2}(N(S) \cup S)$ of these edges originate from vertices of $S' \cup N(S')$. This leaves at least $\frac{h}{4}(N(S) \cup S)$ edges available for good vertices, implying that the expected contribution of a scanned vertex u is at least $\frac{h}{4n}(N(S) \cup S)$. Hence in expectation not more than $\frac{4n}{h}$ vertices are needed until $N(S) \cup S$ doubles its size. As a subphase contains $\frac{8n}{h}$ vertices, item 2 of Lemma 4 implies that the probability that there is an unsuccessful phase is at most $\sum_{\ell \geq 1} \frac{2\Delta}{4\Delta 2^\ell} \leq 1/2$.

□

Our proof for Theorem 4 did not attempt to optimize the value of the leading constant c.

Each of the bounds in Theorems 3 and 4 may be better than the other, depending on the relative values of n, Δ, h. In our intended applications $\frac{\Delta}{h}$ is much smaller than $\frac{n}{\Delta}$, and hence we shall use Theorem 4.

The requirement that G is regular in Theorems 3 and 4 was made because it simplifies the proofs. This requirement can be removed by slightly adjusting the statement of the theorem.

Theorem 5. *Let $G(V, E)$ be an n-vertex graph with edge expansion at least h, and let Δ denote the degree of the vertex of $\frac{h}{4}$th highest degree (breaking ties arbitrarily). Then with probability at least $1/2$, a random sample U of $\frac{cn}{h}\min[\log\frac{n}{\Delta}, \log\frac{\Delta}{h}]$ vertices contains a subset $S \subset U$ with $|N(S) \cup S| \geq n/2$ whose induced subgraph is connected.*

Proof. Let $H \subset V$ be the set of $h/4$ highest degree vertices in G. We make a preliminary pass over all vertices of U. This pass is successful if $1 \leq |U \cap H| \leq 5c\min[\log\frac{n}{\Delta}, \log\frac{\Delta}{h}]$. This fails with probability $2^{-\Omega(c)}$.

If the preliminary pass achieved its goal, we keep in S only one vertex r chosen arbitrarily from $H \cap S$, and remove from U all other vertices of $H \cap U$. The remaining size of U is at least $(cn/h - 5c)\min[\log\frac{n}{\Delta}, \log\frac{\Delta}{h}]$. This remaining size is still roughly $\frac{cn}{h}\min[\log\frac{n}{\Delta}, \log\frac{\Delta}{h}]$. (This would fail to hold only if h is $\Omega(n)$. However, in that case the proofs of Theorems 3 and 4 apply, requiring only an adjustment of the constants hidden in the c notation.) Observe that for every set $T \subset (V \setminus H)$ of size at most $n/2 - 1$, the set $\{r\} \cup T$ has edge expansion at least $3h/4$ into $V \setminus H$. Replace G by the subgraph G' induced on $V' = (V \setminus H) \cup r$. Observe that in this subgraph the degree of r is at least its

original degree (which was necessarily at least Δ) minus $h/4$. Hence the degree of r in this subgraph is at least $\Delta/2$ (it is not hard to show that $h/4 \leq \Delta/2$), and moreover, no vertex in V' has degree larger than Δ.

It can readily seen that the proofs of Theorems 3 and 4 did not use regularity of G, but rather the following two aspects of Δ: that r has degree at least Δ, and no vertex has degree larger than Δ. For G' the only difference is that the degree of r is at least $\Delta/2$ rather than Δ (and the expansion is also smaller by a constant factor). The proofs of Theorems 3 and 4 apply, requiring only an adjustment of the constants hidden in the c notation. □

We now prove Theorem 2.

Proof. (Theorem 2). Recall that H has $\frac{(1+o(1))n}{h}\ln(\frac{n}{h}) + O\left(\frac{n}{h}\log(\frac{\Delta}{h})\right)$ random vertices. Of them, we use up $O\left(\frac{n}{h}\log(\frac{\Delta}{h})\right)$ random vertices in the proof of Theorem 5, and as a consequence we conclude (with probability that can be made arbitrarily close to 1, by changing the constant in the O notation) that H has a connected component CC_v satisfying $|N(CC_v)\bigcup CC_v| \geq n/2$. This was referred to as Step 1 above. Now we analyze Step 2, which is based on considering the remaining random vertices of H, whose number is at least $\frac{(1+o(1))n}{h}\ln(\frac{n}{h})$.

Let us define a linear order on all vertices of G. The property of this linear order is that for every vertex $u \in V$, either it precedes at least h of its neighbors in this linear order, or $u \in N(CC_v)\bigcup CC_v$. Such a linear order exists by the expansion properties of G. The vertices of $N(CC_v)\bigcup CC_v$ can be placed last in this linear order. As for the set R of remaining vertices, the cardinality of R is at most $n/2$, and hence $e_G(R, V_G \setminus R) \geq h|R|$. Hence at least one vertex in R has h neighbors already placed later than R in the linear order, and this vertex can be placed last among R. This argument can be continued by induction to complete the desired linear order.

Consider now the placement of all vertices of H in this linear order. If for every vertex $u \in H$, either $u \in N(CC_v)\bigcup CC_v$, or there is a vertex $w \in N(u)$ that is also in H and moreover w appears later than u in the linear order, then H is connected (because every vertex has a path to v). In our intended applications $\frac{(1+o(1))n}{h}\ln(\frac{n}{h}) \geq \Omega\left(\frac{n}{h}\log(\frac{\Delta}{h})\right)$, implying that $|H| = O\left(\frac{n}{h}\ln(\frac{n}{h})\right)$. In this case, taking $\frac{(1+o(1))n}{h}\ln(\frac{n}{h})$ random vertices of G in Step 2, a union bound implies that the probability that connectivity fails is at most $|H|e^{-(1+o(1))\ln(n/h)} \leq o(1)$. Note also that even if the condition $\frac{(1+o(1))n}{h}\ln(\frac{n}{h}) \geq \Omega\left(\frac{n}{h}\log(\frac{\Delta}{h})\right)$ does not hold, then in Step 2 we may take $\Omega\left(\frac{n}{h}\log(\frac{\Delta}{h})\right)$ random vertices, and the union bound works as well. □

5 A Note on the Dimension Range

Recall the definition of sphere casps, see Section 2. In this section we use the same bounds as in [7]:

Theorem 6. *(Bounds on the sphere Cap measure).* $\frac{c}{\sqrt{d}}(1-a^2)^{\frac{d-1}{2}} \leq |C_a| \leq \frac{1}{2}(1-a^2)^{\frac{d-1}{2}}$, *where c is some constant independent of d.*

Assume that the dimension d of our graphs $G_{n,\pi/3,d}$ is $c\ln(n)$. It follows that when c gets larger each vertex has fewer neighbors. We would like to determine the values of c for which $G \in_R G_{n,\pi/3,d}$ has isolated vertices with high probability. Let v be a vertex in G it holds that $E[|N(v)|] = (n-1)\left|C_{\frac{1}{2}}\right|$. Theorem 6 implies $\left|C_{\frac{1}{2}}\right|$ can be upper bounded by $\frac{1}{2}\left(1-\frac{1}{2}^2\right)^{\frac{d-1}{2}}$. Therefore the expected number of neighbors of each vertex can be upper bounded:

$$(n-1)\left|C_{\frac{1}{2}}\right| \leq \frac{1}{2}n\left(1-\left(\frac{1}{2}\right)^2\right)^{\frac{d-1}{2}} = \frac{1}{2}ne^{\ln\left(\frac{3}{4}\right)\frac{d-1}{2}}$$

$$= O(1)\,ne^{\ln\left(\frac{3}{4}\right)\frac{d}{2}} = O(1)\,n^{1+\ln\left(\frac{3}{4}\right)\frac{c}{2}}$$

Therefore if $1+\ln\left(\frac{3}{4}\right)\frac{c}{2} < 0 \Rightarrow c > \frac{2}{\ln\left(\frac{4}{3}\right)} = 6.95212$ then (by applying the Markov's inequality) the probability that v is isolated tends to one.

Now we determine the values of c for which $G \in_R G_{n,\pi/3,d}$ has no isolated vertices with high probability. Note that by Fact 6 $\left|C_{\frac{1}{2}}\right|$ can be lower bounded by $\frac{O(1)}{\sqrt{d}}\left(1-\frac{1}{2}^2\right)^{\frac{d-1}{2}}$. Therefore the expected number of neighbors of each vertex can be lower bounded:

$$(n-1)\left|C_{\frac{1}{2}}\right| \geq \frac{O(1)}{\sqrt{d}}n\left(1-\left(\frac{1}{2}\right)^2\right)^{\frac{d-1}{2}} = \frac{O(1)}{\sqrt{d}}n^{1+\ln\left(\frac{3}{4}\right)\frac{c}{2}}$$

Choose c so that the right hand side is somewhat larger than $\ln n$. This requires $1+\ln\left(\frac{3}{4}\right)\frac{c}{2}$ to be slightly larger than 0, which happens for $c \simeq \frac{2}{\ln\left(\frac{4}{3}\right)} = 6.95212$. Then standard large deviation bounds imply each vertex has probability $o(\frac{1}{n})$ of having no neighbors, and then by applying the union bound the probability that there is an isolated vertex is $o(1)$.

6 More on Vertex Percolation in Expander Graphs

We say that a graph G has a *majority component* if G has a connected component containing at least half its vertices. The following corollary is not needed in order to prove Theorem 1, but might be of independent interest.

Corollary 1. *Let $G(V,E)$ be an n-vertex Δ-regular graph with edge expansion at least h. Consider $G[U]$, a subgraph of G induced on a random sample U of $\frac{cn}{h}\log\frac{\Delta}{h}$ vertices. Then with probability at least $1/2$ over the choice of U, the subgraph $G[U]$ contains a majority component.*

Proof. Set the value of c to be large enough so that a simple adaptation of the proof of Theorem 4 implies that every vertex $u \in U$ has probability at least

9/10 of being in a component S_u with $|S_u \cup N(S_u)| > (n+h)/2$. Observe that if $|S_u \cup N(S_u)| > (n+h)/2$ and $|S_v \cup N(S_v)| > (n+h)/2$ then $|(S_u \cup N(S_u)) \cap (S_v \cup N(S_v))| \geq h$. Moreover, any vertex of U that lies in $(S_u \cup N(S_u)) \cap (S_v \cup N(S_v))$ connects S_u and S_v.

Fix one arbitrary vertex $u \in U$ and analyse $|S_u|$. With probability at least 9/10 we have $|S_u \cup N(S_u)| > (n+h)/2$. For every $v \neq u$, $v \in U$, we also have probability at least 9/10 to have $|S_v \cup N(S_v)| > (n+h)/2$. If both events hold, then with overwhelming probability v is in the same component as u (one can reserve a small fraction of the vertices of U specifically for the purpose of checking whether they land in $(S_u \cup N(S_u)) \cap (S_v \cup N(S_v))$). Hence given that $|S_u \cup N(S_u)| > (n+h)/2$, the expected number of vertices not in S_u is at most roughly $|U|/10$, implying that with probability at most roughly 1/5 it exceeds $|U|/2$. Hence the probability that $|S_u| < |U|/2$ is at most $\frac{1}{5} + \frac{1}{10} + \epsilon < \frac{1}{2}$, where the ϵ term accounts for low probability events ignored n the computation. □

Acknowledgements. Work 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). We thank Gideon Schechtman for helpful discussions.

References

1. Alon, N., Benjamini, I., Stacey, A.: Percolation on finite graphs and isoperimetric inequalities. Annals of Probability, 1727–1745 (2004)
2. Appel, M.J.B., Russo, R.P.: The connectivity of a graph on uniform points on $[0,1]^d$. Statistics & Probability Letters 60(4), 351–357 (2002)
3. Ben-Shimon, S., Krivelevich, M.: Vertex percolation on expander graphs. European Journal of Combinatorics 30(2), 339–350 (2009)
4. David, R.: Finding planted k-coloring in vector k-colorable graphs. MSC Thesis, Weizmann Institute (2013), `http://www.wisdom.weizmann.ac.il/ feige/ TechnicalReports/RoeeDavidThesis.pdf`
5. Erdos, P., Renyi, A.: On random graphs I. Publ. Math. Debrecen 6, 290–297 (1959)
6. Feige, Langberg, Schechtman: Graphs with tiny vector chromatic numbers and huge chromatic numbers. SICOMP: SIAM Journal on Computing 33 (2004)
7. Feige, U., Schechtman, G.: On the optimality of the random hyperplane rounding technique for max cut. Random Structures & Algorithms 20(3), 403–440 (2002)
8. Godehardt, E., Jaworski, J.: On the connectivity of a random interval graph. Random Structures & Algorithms 9(1-2), 137–161 (1996)
9. Goel, A., Rai, S., Krishnamachari, B.: Sharp thresholds for monotone properties in random geometric graphs. In: Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing, pp. 580–586. ACM (2004)
10. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM (JACM) 42(6), 1115–1145 (1995)
11. Grimmett, G.: What is Percolation? Springer (1999)
12. Gupta, P., Kumar, P.R.: Critical power for asymptotic connectivity in wireless networks. In: Stochastic Analysis, Control, Optimization and Applications, pp. 547–566. Springer (1998)

13. Karger, D., Motwani, R., Sudan, M.: Approximate graph coloring by semidefinite programming. Journal of the ACM (JACM) 45(2), 246–265 (1998)
14. Li, S.: Concise formulas for the area and volume of a hyperspherical cap. Asian Journal of Mathematics and Statistics 4(1), 66–70 (2011)
15. Meester, R., Roy, R.: Continuum percolation. Cambridge tracts in mathematics, vol. 119 (1996)
16. Penrose, M.D.: Random geometric graphs (2003)
17. Penrose, M.D.: Continuum percolation and euclidean minimal spanning trees in high dimensions. The Annals of Applied Probability 6(2), 528–544 (1996)
18. Penrose, M.D.: The random minimal spanning tree in high dimensions. The Annals of Probability 24(4), 1903–1925 (1996)
19. Penrose, M.D.: The longest edge of the random minimal spanning tree. Ann. Appl. Probab. 7(2), 340–361 (1997)

Matching-Vector Families and LDCs over Large Modulo

Zeev Dvir[1,⋆] and Guangda Hu[2,⋆]

[1] Department of Computer Science and Department of Mathematics, Princeton University
zeev.dvir@gmail.com
[2] Department of Computer Science, Princeton University
guangdah@cs.princeton.edu

Abstract. We prove new upper bounds on the size of families of vectors in $\mathbb{Z}_m^n$ with restricted modular inner products, when m is a large integer. More formally, if $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_t \in \mathbb{Z}_m^n$ and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t \in \mathbb{Z}_m^n$ satisfy $\langle \boldsymbol{u}_i, \boldsymbol{v}_i \rangle \equiv 0 \pmod m$ and $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle \not\equiv 0 \pmod m$ for all $i \neq j \in [t]$, we prove that $t \leq O(m^{n/2+8.47})$. This improves a recent bound of $t \leq m^{n/2+O(\log(m))}$ by [BDL13] and is the best possible up to the constant 8.47 when m is sufficiently larger than n.

The maximal size of such families, called 'Matching-Vector families', shows up in recent constructions of locally decodable error correcting codes (LDCs) and determines the rate of the code. Using our result we are able to show that these codes, called Matching-Vector codes, must have encoding length at least $K^{19/18}$ for K-bit messages, regardless of their query complexity. This improves a known super linear bound of $K2^{\Omega(\sqrt{\log K})}$ proved in [DGY11].

1 Introduction

A Matching-Vector family (MV family) in $\mathbb{Z}_m^n$ is defined as a pair of ordered lists $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_t)$ and $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$ with $\boldsymbol{u}_i, \boldsymbol{v}_j \in \mathbb{Z}_m^n$, satisfying the following property: for all $i \in [t]$, $\langle \boldsymbol{u}_i, \boldsymbol{v}_i \rangle \equiv 0 \pmod m$ whereas for all $i \neq j \in [t]$, $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle \not\equiv 0 \pmod m$. Here $\langle \cdot, \cdot \rangle$ denotes the standard inner product. If one restricts the entries of the vectors in the family to be in the set $\{0, 1\}$ the inner products corresponds to the sizes of the intersections (modulo m) and, in this case, MV families are more commonly referred to as families of sets with restricted modular intersections. MV families were studied previously in the context of Ramsey graphs [Gro00], circuit complexity [BBR94] and, more recently, were used to construct Locally Decodable Codes (LDCs) [Yek08, Efr09, DGY11], which are error correcting codes with super-efficient decoding properties. We will elaborate more on the connection to LDCs after we state our results.

We denote by $\mathrm{MV}(m, n)$ the size of the largest MV family in $\mathbb{Z}_m^n$ (the size of the family is t in the above notation). It is an interesting (and mostly open) question

⋆ Research partially supported by NSF grants CCF-0832797, CCF-1217416 and by the Sloan fellowship.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 513–526, 2013.

to determine the value (or even order of magnitude) of $\mathrm{MV}(m,n)$ for arbitrary m and n. Upper and lower bounds on $\mathrm{MV}(m,n)$ can be roughly divided into two kinds, corresponding to the relative size of the two parameters. One typical regime is when m is small and n tends to infinity and the other is when $m >> n$ (of course there are intermediate scenarios as well).

Although our work focuses on the regime when m is much larger than n, we first describe the known results for the other regime, namely when m is a fixed constant and n tends to infinity. These regime is further divided into the case when m is prime and when m is composite. When m is a small *prime* and n tends to infinity, the value of $\mathrm{MV}(m,n)$ is known to be of the order of n^{m-1} [BF98]. When m is a small *composite*, the picture is very different and there are exponential gaps between known lower and upper bounds on $\mathrm{MV}(m,n)$. A surprising construction by Grolmuzs [Gro00] shows that $\mathrm{MV}(m,n) \geq \exp\left(c \cdot \frac{(\log n)^r}{(\log\log n)^{r-1}}\right)$ when m has r distinct prime factors (here c is an absolute constant). That is, $\mathrm{MV}(m,n)$ can be super-polynomial in n (that is $n^{\omega(1)}$) for m as small as 6 (compared with the polynomial upper bound n^{m-1} for prime m). A trivial upper bound on $\mathrm{MV}(m,n)$ is m^n since an MV family cannot contain the same vector twice. The best upper bound on $\mathrm{MV}(m,n)$ for small composite m was proved in [BDL13] and is $m^{n/2+O(\log m)}$. Assuming the Polynomial-Freiman-Ruzsa (PFR) conjecture [TV07] this can be improved to $\mathrm{MV}(m,n) \leq C_m^{n/\log n}$ with C_m a constant depending only on m.

Table 1. List of upper bounds on $\mathrm{MV}(m,n)$

m	upper bound for $\mathrm{MV}(m,n)$
general prime	$O(m^{n/2})$ [DGY11]
small, fixed prime	$O(n^{m-1})$ [BF98]
general composite	$m^{n/2+O(\log m)}$ [BDL13]
small, fixed composite	$2^{O_m(n/\log n)}$ [BDL13] (assuming PFR)
general composite	$O(m^{n/2+8.47})$ (Theorem 1.1)

Our work focuses on the regime when m is larger than n. In this setting, a construction of [YGK12] gives MV families of size $\left(\frac{m+1}{n-2}\right)^{n/2-1}$ [YGK12]. For a large *prime* m, this construction almost matches an upper bound of $O(m^{n/2})$ proved in [DGY11]. For composite m, the best upper bound on $\mathrm{MV}(m,n)$ for large m prior to this work was the same $m^{n/2+O(\log m)}$ bound from [BDL13]. Notice that, when $m > 2^n$, this bound is meaningless since it exceeds the trivial bound of m^n. In this work we extend the proof method developed in [BDL13] to give the following bound:

Theorem 1.1. *For all integers $m > 1$ and $n > 0$, we have $MV(m,n) \leq 100m^{n/2+8.47}$. When m is a product of distinct primes the constant 8.47 can be replaced with $4 + o(1)$.*

For small n, this bound is tight up to the constant 8.47 as the [YGK12] construction shows. When m is small, this still gives some improvement over the $m^{n/2+O(\log m)}$ bound of [BDL13] but not as dramatic (and probably far from being tight).

The main tool in our proof is Fourier analysis in the spirit of [BDL13], with which we repeatedly reduce m to one of its factor (eventually reaching the case of $m = 1$). The distribution of $\langle \boldsymbol{v_i}, \boldsymbol{u_j} \rangle$ over random $i, j \in [t]$ is far from the uniform distribution (since the probability of obtaining zero is small). This fact is used to find a large coefficient in its Fourier spectrum. This coefficient is then used to carve out a large sub family which is again an MV family, but over some proper factor of m. The proof ends when we reach the case of prime m. The difference between our proof and the one in [BDL13] is in the choice of the large coefficient (or character). We are able to show that a large character appears that has nicer number theoretic properties and so are able to analyze the loss in each step in a better way – getting rid of the $O(\log m)$ factor in the exponent.

1.1 MV Families and Locally Decodable Codes

A (q, δ, ϵ)-Locally Decodable Code, or LDC, encodes a K-symbol message x to an N-symbol codeword $C(x)$, such that every symbol x_i ($i \in [K]$) can be recovered with probability at least $1 - \epsilon$ by a randomized decoding procedure that makes only q queries to $C(x)$, even if δN locations of the codeword $C(x)$ have been corrupted. Understanding the minimum length $N = N(k)$ of an LDC with constant q is a central research question that is still far from being solved. For $q = 1, 2$, this question is completely answered. There are no LDCs for $q = 1$ [KT00] and the best LDCs for $q = 2$ have exponential length [GKST02, KdW04]. However, for $q > 2$ there are huge gaps between lower bound and LDC constructions. The best known lower bound is $N = \tilde{\Omega}(K^{1+1/(\lceil r/2 \rceil - 1)})$ for $k \geq 4$ [KdW04, Woo07] and $N = \Omega(K^2)$ for $k = 3$ [Woo10], while the best construction has super-polynomial length. Constructions of LDCs have been studied extensively for more than a decade. Until recently, all constructions of LDC with constant q had exponential encoding length. In a breakthrough work of Yekhanin [Yek08] and following improvements [Efr09, Rag07, KY09, IS10, CFL$^+$10, DGY11, BET10], a new family of LDCs based on Matching Vector families was introduced. These codes, called Matching-Vector codes, rely on constructions of MV families and can have sub-exponential length for q as small as 3 [Efr09]. Using Grolmuzs construction as a building block, one obtains an encoding length of roughly

$$N \sim \exp\exp\left((\log K)^{O(\log\log q/\log q)}(\log\log K)\right).$$

The size of the MV family used in the code construction is critical. In its simplest form, an MV code using an MV family of size t in $\mathbb{Z}_m^n$ will send $K = t$ bits of message into $N = m^n$ bits of encoding and will require $q = m$ queries to decode. Several improvements are possible for reducing the number of queries below m but these are case-based and hard to generalize for arbitrary m.

Our improved bound on the size of MV families allows us to prove an unconditional lower bound on the encoding length of MV codes, *regardless of the query complexity.*

Theorem 1.2. *For any MV-code with message length K and codeword length N we have $N > K^{\frac{19}{18}}$. This bound is regardless of the number of queries.*

This theorem improves on a bound of $N > K2^{\Omega(\sqrt{\log K})}$ proved in [DGY11].

1.2 Organization

We begin in Section 2 with a number of preliminary lemmas and notations that will be used throughout the proof. In Section 3 we prove our main technical lemma which is the heart of our proof. The lemma is used iteratively in the proof of our main theorem which is given in Section 4. The proof of the stronger bound for the case when m is a product of distinct primes is given in Section 5.

2 Preliminaries

2.1 Fourier Lemma

We consider a probability distribution μ over $\mathbb{Z}_m$. Let $\omega_m = e^{\frac{2\pi}{m}i}$ be an order m primitive root of unity. It is not difficult to see $\mathbb{E}_{x\sim\mu}[\omega_m^{jx}] = 0$ for all $j \in \{1, 2, \ldots, m-1\}$ if μ is the uniform distribution. We will show that $\mathbb{E}_{x\sim\mu}[\omega_m^{jx}]$ is bounded away from zero for some $j \in \{1, 2, \ldots, m-1\}$ if μ is far from being uniform.

In [BDL13], it was shown that

$$\max_{1\le j\le m-1} \left| \mathop{\mathbb{E}}_{x\sim\mu}[\omega_m^{jx}] \right| = \Omega(\frac{1}{m^{1.5}})$$

if the statistical distance between μ and the uniform distribution is big, i.e. $\frac{1}{2}\sum_{x\in\mathbb{Z}_m} |\mu(x) - \frac{1}{m}| = \Omega(\frac{1}{m})$. In the following lemma, we prove a better lower bound that depends only on $s = \text{order}(\omega_m^j)$ under a stronger condition $|\mu(0) - \frac{1}{m}| = \Omega(\frac{1}{m})$.

Consider μ as a function from $\mathbb{Z}_m$ to $\mathbb{C}$. For $0 \le j \le m-1$, the Fourier coefficient $\hat{\mu}(j)$ is

$$\hat{\mu}(j) = \frac{1}{m}\sum_{x\in\mathbb{Z}_m} \mu(x)\omega_m^{-jx} = \frac{1}{m}\mathop{\mathbb{E}}_{x\sim\mu}[\omega_m^{-jx}].$$

One can see that $\hat{\mu}(0) = \frac{1}{m}$. The set of functions $\{\omega_m^{jx} \mid 0 \le j \le m-1\}$ is an orthogonal basis for all functions from $\mathbb{Z}_m$ to $\mathbb{C}$, and the function $\mu(x)$ can be written as

$$\mu(x) = \sum_{j=0}^{m-1} \hat{\mu}(j)\omega_m^{jx}. \tag{1}$$

Lemma 2.1. *Let $\mu : \mathbb{Z}_m \mapsto [0,1]$ be a probability distribution over $\mathbb{Z}_m$ (i.e. $\sum_{x\in\mathbb{Z}_m} \mu(x) = 1$). If $\mu(0) \le \frac{1}{100m}$, there must exist $j \in \{1,2,\ldots,m-1\}$ such that $\left|\mathbb{E}_{x\sim\mu}[\omega_m^{jx}]\right| \ge \frac{1}{sf(s)}$, where $s = \frac{m}{\gcd(j,m)}$ is the order of ω_m^j for $\omega_m = e^{\frac{2\pi}{m}i}$, and $f : [2,\infty) \mapsto \mathbb{R}^+$ is any function satisfying $\sum_{s=2}^{\infty} \frac{1}{f(s)} \le 0.99$.*

Proof. By setting $x = 0$ in (1), we have

$$\mu(0) = \sum_{j=0}^{m-1} \hat{\mu}(j)\omega_m^{j\cdot 0} = \sum_{j=0}^{m-1} \hat{\mu}(j) = \frac{1}{m} + \frac{1}{m}\sum_{j=1}^{m-1} \mathop{\mathbb{E}}_{x\sim\mu}[\omega_m^{-jx}].$$

Therefore

$$\begin{aligned}\sum_{j=1}^{m-1} \left|\mathop{\mathbb{E}}_{x\sim\mu}[\omega_m^{jx}]\right| &\ge \left|\sum_{j=1}^{m-1} \mathop{\mathbb{E}}_{x\sim\mu}[\omega_m^{jx}]\right| = \left|\sum_{j=1}^{m-1} \mathop{\mathbb{E}}_{x\sim\mu}[\omega_m^{-jx}]\right| \\ &= m\cdot\left|\mu(0) - \frac{1}{m}\right| \ge 0.99. \end{aligned} \tag{2}$$

The first equality is because $\omega_m^j = \omega_m^{m-(m-j)} = \omega_m^{-(m-j)}$ and when j takes values $\{1,2,\ldots,m-1\}$, $m-j$ also takes each of these values.

For every $d \mid m$ $(1 \le d \le m-1)$, define $T_d = \{j \mid \gcd(j,m) = d, 1 \le j \le m-1\}$. For all $j \in T_d$, the order of ω_m^j is $s_d = \frac{m}{d}$ $(2 \le s_d \le m)$. We also see $T_d = \{k\cdot d \mid 1 \le k < s_d, \gcd(k,s_d) = 1\}$, hence $|T_d| = \varphi(s_d) < s_d$.

If the lemma was not true, we have

$$\begin{aligned}\sum_{j=1}^{m-1} \left|\mathop{\mathbb{E}}_{x\sim\mu}[\omega_m^{jx}]\right| &= \sum_{\substack{d\mid m\\ d<m}} \left(\sum_{j\in T_d} \left|\mathop{\mathbb{E}}_{x\sim\mu}[\omega_m^{jx}]\right|\right) \\ &< \sum_{\substack{d\mid m\\ d<m}} \left(s_d \cdot \frac{1}{s_d f(s_d)}\right) < \sum_{s=2}^{\infty} \frac{1}{f(s)} \le 0.99.\end{aligned}$$

This violates inequality (2). Thus the lemma is proved. □

2.2 Notations and Facts about MV Families

We use $\langle\cdot,\cdot\rangle$ to denote the inner product over $\mathbb{Z}$ between two vectors. In all calculations, we identify $\mathbb{Z}_m$ as $\{0,1,\ldots,m-1\}$ and treat the numbers as on $\mathbb{Z}$. Conventionally, we consider a mod 1 to be 0 for any integer a.

Notation 2.2. *Let r be a positive integer. For an integer v, define $v^{(r)} \in \{0,1,\ldots,r-1\}$ to be v modulo r. For a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$, define $\boldsymbol{v}^{(r)} = (v_1^{(r)}, v_2^{(r)}, \ldots, v_n^{(r)})$. For a list of vectors $V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_t)$, define $V^{(r)} = (\boldsymbol{v}_1^{(r)}, \boldsymbol{v}_2^{(r)}, \ldots, \boldsymbol{v}_t^{(r)})$.*

Notation 2.3. *Let r be a positive integer. For an integer v, define $v^{[r]} \in \mathbb{Z}$ to be $(v - v^{(r)})/r$. For a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$, define $\boldsymbol{v}^{[r]} = (v_1^{[r]}, v_2^{[r]}, \ldots, v_n^{[r]})$. Thus $\boldsymbol{v} = r\boldsymbol{v}^{[r]} + \boldsymbol{v}^{(r)}$ for any vector $\boldsymbol{v}$. For a list of vectors $V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_t)$, define $V^{[r]} = (\boldsymbol{v}_1^{[r]}, \boldsymbol{v}_2^{[r]}, \ldots, \boldsymbol{v}_t^{[r]})$.*

Definition 2.4. *Let $U = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_t)$ and $V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_t)$ be two lists of vectors in $\mathbb{Z}_m^n$. (U, V) is a* matching vector family *if $\langle \boldsymbol{u}_i, \boldsymbol{v}_i \rangle \equiv 0 \pmod m$ for all $i \in [t]$ and $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle \not\equiv 0 \pmod m$ for all $i \neq j \in [t]$. The number t is the* size *of the MV family and is denoted by $|(U, V)|$.*

Claim 2.5. *For an MV family (U, V) where $U = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_t)$, $V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_t)$ and $i \neq j \in [t]$, we have $\boldsymbol{u}_i \neq \boldsymbol{u}_j$ and $\boldsymbol{v}_i \neq \boldsymbol{v}_j$.*

Proof. Assume $\boldsymbol{u}_i = \boldsymbol{u}_j$ for $i \neq j$, we have $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle = \langle \boldsymbol{u}_i, \boldsymbol{v}_i \rangle \equiv 0 \pmod m$. This violates the definition of MV family. □

Notation 2.6. *Let U, V, U', V' be 4 lists of vectors in $\mathbb{Z}_m^n$, and say $U = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_t)$, $V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_t)$. We write $(U', V') \subseteq (U, V)$ if there exists a set $T \subseteq [t]$ such that $U' = (\boldsymbol{u}_i : i \in T)$ and $V' = (\boldsymbol{v}_i : i \in T)$. Observe that if (U, V) is an MV family, so is (U', V').*

Definition 2.7. *(r_1, r_2, r_3) is a* partition *of m if $r_1, r_2, r_3 \in \mathbb{Z}^+$ and $r_1 r_2 r_3 = m$. (r_1, r_2, r_3 are not assumed to be coprime.)*

Definition 2.8. *For an MV family (U, V) where $U = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_t)$, $V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_t)$, we say (U, V)* respects *(r_1, r_2, r_3), where (r_1, r_2, r_3) is a partition of m, if the following conditions are satisfied:*

1. *$\exists \boldsymbol{u}_0 \in \mathbb{Z}_{r_1}^n$ such that $\boldsymbol{u}_i^{(r_1)} = \boldsymbol{u}_0$ for all $i \in [t]$,*
2. *$\exists \boldsymbol{v}_0 \in \mathbb{Z}_{r_2}^n$ such that $\boldsymbol{v}_i^{(r_2)} = \boldsymbol{v}_0$ for all $i \in [t]$,*
3. *$\langle \boldsymbol{u}_i^{[r_1]}, \boldsymbol{v}_0 \rangle$ modulo r_2 is the same for all $i \in [t]$,*
4. *$\langle \boldsymbol{u}_0, \boldsymbol{v}_i^{[r_2]} \rangle$ modulo r_1 is the same for all $i \in [t]$.*

Claim 2.9. *If an MV family (U, V) respects (r_1, r_2, r_3), then $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle \equiv 0 \pmod{r_1 r_2}$ for all $\boldsymbol{u}_i \in U, \boldsymbol{v}_j \in V$.*

Proof. Let $\boldsymbol{u}_0 = \boldsymbol{u}_i^{(r_1)}$ and $\boldsymbol{v}_0 = \boldsymbol{v}_j^{(r_2)}$. They are fixed for all $\boldsymbol{u}_i \in U$ and $\boldsymbol{v}_j \in V$. We have

$$\begin{aligned}\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle &= \langle r_1 \boldsymbol{u}_i^{[r_1]} + \boldsymbol{u}_0, r_2 \boldsymbol{v}_j^{[r_2]} + \boldsymbol{v}_0 \rangle \\ &= r_1 r_2 \langle \boldsymbol{u}_i^{[r_1]}, \boldsymbol{v}_j^{[r_2]} \rangle + r_1 \langle \boldsymbol{u}_i^{[r_1]}, \boldsymbol{v}_0 \rangle + r_2 \langle \boldsymbol{u}_0, \boldsymbol{v}_j^{[r_2]} \rangle + \langle \boldsymbol{u}_0, \boldsymbol{v}_0 \rangle.\end{aligned}$$

The first term is 0 modulo $r_1 r_2$. The second term is fixed modulo $r_1 r_2$ because $\langle \boldsymbol{u}_i^{[r_1]}, \boldsymbol{v}_0 \rangle$ is fixed modulo r_2. Similarly, the third term is also a constant modulo $r_1 r_2$. Therefore $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle$ modulo $r_1 r_2$ is the same for all $\boldsymbol{u}_i \in U$ and $\boldsymbol{v}_j \in V$. Note that when $i = j$, $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle \equiv 0 \pmod{r_1 r_2}$ since (U, V) is an MV family. Therefore $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle \equiv 0 \pmod{r_1 r_2}$ for all $\boldsymbol{u}_i \in U, \boldsymbol{v}_j \in V$. □

Claim 2.10. *Every MV family (U, V) respects $(1, 1, m)$.*

Proof. Let $\boldsymbol{u}_0$ and $\boldsymbol{v}_0$ be the zero vector. All the conditions are satisfied. □

Claim 2.11. *If an MV family (U, V) respects $(r_1, r_2, 1)$, then it must has size 1.*

Proof. Since $r_1 r_2 = m$, by Claim 2.9 we have $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle \equiv 0 \pmod m$ for all $\boldsymbol{u}_i \in U, \boldsymbol{v}_j \in V$. By the definition of MV family, the size of (U, V) must be 1. □

3 Proof of the Main Lemma

Consider an MV family (U, V), where $U = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_t)$ and $V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_t)$. We pick $\boldsymbol{u} \in U$ and $\boldsymbol{v} \in V$ uniformly at random and consider the distribution of $\langle \boldsymbol{u}, \boldsymbol{v} \rangle^{(m)}$. The inner product is 0 with probability $1/t$. Thus the distribution is far from uniform when $t >> m$. We will take advantage of this fact and prove our key lemma. For an MV family (U, V) respecting (r_1, r_2, r_3), we can find a large subfamily and reduce r_3 to some smaller number.

Let $f : [2, \infty) \mapsto \mathbb{R}^+$ be a function satisfying $\sum_{s=2}^{\infty} \frac{1}{f(s)} \leq 0.99$. We will specify $f(s)$ in later proofs.

Lemma 3.1. *If an MV family (U, V) respects (r_1, r_2, r_3) for some $r_3 \geq 2$ and $|(U, V)| = t \geq 100m$, then there exists $s \mid r_3$ with $s \geq 2$ and an MV family $(U', V') \subseteq (U, V)$ with $|(U', V')| \geq t/(s^{n/2+4} f(s)^2)$ that respects either $(r_1 s, r_2, r_3/s)$ or $(r_1, r_2 s, r_3/s)$.*

Proof. We prove the lemma in 4 steps.

Step 1: Finding a character with a large bias. By Claim 2.9, $\frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{r_1 r_2}$ is an integer for all $\boldsymbol{u} \in U, \boldsymbol{v} \in V$. We can also see $\frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{r_1 r_2} \equiv 0 \pmod{r_3}$ iff $\langle \boldsymbol{u}, \boldsymbol{v} \rangle \equiv 0 \pmod m$. Consider the distribution of $\left(\frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{r_1 r_2} \right)^{(r_3)} \in \mathbb{Z}_{r_3}$, where $\boldsymbol{u}$ and $\boldsymbol{v}$ are uniformly drawn from U and V respectively. We have

$$\begin{aligned} \mathbf{Pr}\left[\left(\frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{r_1 r_2} \right)^{(r_3)} = 0 \right] &= \mathbf{Pr}\Big[\langle \boldsymbol{u}, \boldsymbol{v} \rangle \equiv 0 \pmod m \Big] \\ &= \frac{1}{t} \leq \frac{1}{100m} \leq \frac{1}{100 r_3}. \end{aligned}$$

Applying Lemma 2.1 on $\mathbb{Z}_{r_3}$, there exists a $j \in \{1, 2, \ldots, r_3 - 1\}$ such that

$$\left| \mathop{\mathbb{E}}_{\substack{\boldsymbol{u} \sim U \\ \boldsymbol{v} \sim V}} \left[\omega_{r_3}^{j \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{r_1 r_2}} \right] \right| \geq \frac{1}{s f(s)}, \tag{3}$$

where $\omega_{r_3} = e^{\frac{2\pi i}{r_3}}$ and $s = \frac{r_3}{\gcd(j,r_3)}$ is the order of $\omega_{r_3}^j$. Note that we have dropped the modulo r_3 operation because $(\omega_{r_3}^j)^{r_3} = 1$. It follows that

$$\mathop{\mathbb{E}}_{\substack{\boldsymbol{u},\widetilde{\boldsymbol{u}}\sim U\\ \boldsymbol{v}\sim V}}\left[\omega_{r_3}^{j\frac{\langle \boldsymbol{u}-\widetilde{\boldsymbol{u}},\boldsymbol{v}\rangle}{r_1 r_2}}\right] = \mathop{\mathbb{E}}_{\boldsymbol{v}\sim V}\left|\mathop{\mathbb{E}}_{\boldsymbol{u}\sim U}\left[\omega_{r_3}^{j\frac{\langle \boldsymbol{u},\boldsymbol{v}\rangle}{r_1 r_2}}\right]\right|^2 \geq \left|\mathop{\mathbb{E}}_{\substack{\boldsymbol{u}\sim U\\ \boldsymbol{v}\sim V}}\left[\omega_{r_3}^{j\frac{\langle \boldsymbol{u},\boldsymbol{v}\rangle}{r_1 r_2}}\right]\right|^2 \geq \frac{1}{s^2 f(s)^2}.$$

Therefore there exists a fixed $\widetilde{\boldsymbol{u}} \in U$ such that

$$\left|\mathop{\mathbb{E}}_{\substack{\boldsymbol{u}\sim U\\ \boldsymbol{v}\sim V}}\left[\omega_{r_3}^{j\frac{\langle \boldsymbol{u}-\widetilde{\boldsymbol{u}},\boldsymbol{v}\rangle}{r_1 r_2}}\right]\right| = \left|\mathop{\mathbb{E}}_{\substack{\boldsymbol{u}\sim U\\ \boldsymbol{v}\sim V}}\left[\omega_{r_3}^{j\langle \frac{\boldsymbol{u}-\widetilde{\boldsymbol{u}}}{r_1},\boldsymbol{v}\rangle/r_2}\right]\right| \geq \frac{1}{s^2 f(s)^2}.$$

Since $\boldsymbol{u}^{(r_1)} = \widetilde{\boldsymbol{u}}^{(r_1)}$, we have $\boldsymbol{u} - \widetilde{\boldsymbol{u}} = r_1(\boldsymbol{u}^{[r_1]} - \widetilde{\boldsymbol{u}}^{[r_1]})$. The above inequality can be written as

$$\left|\mathop{\mathbb{E}}_{\substack{\boldsymbol{u}\sim U\\ \boldsymbol{v}\sim V}}\left[\omega_{r_3}^{j\langle \boldsymbol{u}^{[r_1]}-\widetilde{\boldsymbol{u}}^{[r_1]},\boldsymbol{v}\rangle/r_2}\right]\right| \geq \frac{1}{s^2 f(s)^2}. \tag{4}$$

Step 2: Partitioning into buckets. We partition the set U into buckets according to $\boldsymbol{u}^{[r_1]} - \widetilde{\boldsymbol{u}}^{[r_1]}$ modulo s: $U = \bigcup_{\boldsymbol{w}\in\mathbb{Z}_s^n} B(\boldsymbol{w}, U)$, where

$$\widetilde{B}(\boldsymbol{w}, U) = \left\{\boldsymbol{u} \in U \;\middle|\; \left(\boldsymbol{u}^{[r_1]} - \widetilde{\boldsymbol{u}}^{[r_1]}\right)^{(s)} = \boldsymbol{w}\right\}.$$

We also partition V into buckets $B(\boldsymbol{w}, V) = \{\boldsymbol{v} \in V \mid (\boldsymbol{v}^{[r_2]})^{(s)} = \boldsymbol{w}\}$ for all $\boldsymbol{w} \in \mathbb{Z}_s^n$. Define $p_{\boldsymbol{w}} = |\widetilde{B}(\boldsymbol{w}, U)|/t$ to be the density of $\widetilde{B}(\boldsymbol{w}, U)$ and $q_{\boldsymbol{w}} = |B(\boldsymbol{w}, V)|/t$ be the density of $B(\boldsymbol{w}, V)$.

Picking $\boldsymbol{u}$ uniformly from U can be equivalently considered as two steps: 1. For each bucket $\widetilde{B}(\boldsymbol{w}, U)$, pick a representative $\boldsymbol{u}_{\boldsymbol{w}} \in \widetilde{B}(\boldsymbol{w}, U)$ uniformly; 2. Pick one bucket according to the probability distribution $p_{\boldsymbol{w}}$, and output the representative. For inequality (4), we split the procedure of picking $\boldsymbol{u} \sim U$ into these two steps.

$$\begin{aligned}
\frac{1}{s^2 f(s)^2} &\leq \left|\mathop{\mathbb{E}}_{\substack{\boldsymbol{u}\sim U\\ \boldsymbol{v}\sim V}}\left[\omega_{r_3}^{j\langle \boldsymbol{u}^{[r_1]}-\widetilde{\boldsymbol{u}}^{[r_1]},\boldsymbol{v}\rangle/r_2}\right]\right| \\
&= \left|\mathop{\mathbb{E}}_{\substack{\text{for each } \boldsymbol{w},\\ \boldsymbol{u}_{\boldsymbol{w}}\sim\widetilde{B}(\boldsymbol{w},U)}}\;\mathop{\mathbb{E}}_{\boldsymbol{w}\sim p_{\boldsymbol{w}}}\;\mathop{\mathbb{E}}_{\boldsymbol{v}\sim V}\left[\omega_{r_3}^{j\langle \boldsymbol{u}_{\boldsymbol{w}}^{[r_1]}-\widetilde{\boldsymbol{u}}^{[r_1]},\boldsymbol{v}\rangle/r_2}\right]\right| \\
&\leq \mathop{\mathbb{E}}_{\substack{\text{for each } \boldsymbol{w},\\ \boldsymbol{u}_{\boldsymbol{w}}\sim\widetilde{B}(\boldsymbol{w},U)}}\left|\mathop{\mathbb{E}}_{\boldsymbol{w}\sim p_{\boldsymbol{w}}}\;\mathop{\mathbb{E}}_{\boldsymbol{v}\sim V}\left[\omega_{r_3}^{j\langle \boldsymbol{u}_{\boldsymbol{w}}^{[r_1]}-\widetilde{\boldsymbol{u}}^{[r_1]},\boldsymbol{v}\rangle/r_2}\right]\right|.
\end{aligned}$$

There exists a fixed list of representatives from each bucket ($\boldsymbol{u}_{\boldsymbol{w}} \in B(\boldsymbol{w}, U)$: $\boldsymbol{w} \in \mathbb{Z}_s^n$) such that

$$\frac{1}{s^2 f(s)^2} \leq \left|\mathop{\mathbb{E}}_{\substack{\boldsymbol{w}\sim p_{\boldsymbol{w}}\\ \boldsymbol{v}\sim V}}\left[\omega_{r_3}^{j\langle \boldsymbol{u}_{\boldsymbol{w}}^{[r_1]}-\widetilde{\boldsymbol{u}}^{[r_1]},\boldsymbol{v}\rangle/r_2}\right]\right|. \tag{5}$$

For every $\boldsymbol{w} \in \mathbb{Z}_s^n$ and $\boldsymbol{u} \in B(\boldsymbol{w}, U)$, we use $\boldsymbol{u}'$ to denote the vector $(\boldsymbol{u}^{[r_1]} - \widetilde{\boldsymbol{u}}^{[r_1]})^{[s]}$. Thus

$$\boldsymbol{u}_{\boldsymbol{w}}^{[r_1]} - \widetilde{\boldsymbol{u}}^{[r_1]} = s\boldsymbol{u}'_{\boldsymbol{w}} + (\boldsymbol{u}_{\boldsymbol{w}}^{[r_1]} - \widetilde{\boldsymbol{u}}^{[r_1]})^{(s)} = s\boldsymbol{u}'_{\boldsymbol{w}} + \boldsymbol{w}.$$

Hence inequality (5) can be written as

$$\frac{1}{s^2 f(s)^2} \leq \left| \mathop{\mathbb{E}}_{\substack{\boldsymbol{w} \sim p_{\boldsymbol{w}} \\ \boldsymbol{v} \sim V}} \left[\omega_{r_3}^{j\langle s\boldsymbol{u}'_{\boldsymbol{w}} + \boldsymbol{w}, \boldsymbol{v} \rangle / r_2} \right] \right|. \tag{6}$$

Step 3: Finding a large bucket. By inequality (6),

$$\begin{aligned}
\left(\frac{1}{s^2 f(s)^2}\right)^2 &\leq \left| \sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} \sum_{\boldsymbol{v} \in V} p_{\boldsymbol{w}} \cdot \frac{1}{t} \cdot \omega_{r_3}^{j\langle s\boldsymbol{u}'_{\boldsymbol{w}} + \boldsymbol{w}, \boldsymbol{v} \rangle / r_2} \right|^2 \\
&\leq \left(\sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} p_{\boldsymbol{w}}^2 \right) \cdot \left(\sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} \left| \sum_{\boldsymbol{v} \in V} \frac{1}{t} \cdot \omega_{r_3}^{j\langle s\boldsymbol{u}'_{\boldsymbol{w}} + \boldsymbol{w}, \boldsymbol{v} \rangle / r_2} \right|^2 \right) \\
&= \left(\sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} p_{\boldsymbol{w}}^2 \right) \cdot \left(\sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} \sum_{\boldsymbol{v}, \widetilde{\boldsymbol{v}} \in V} \frac{1}{t^2} \cdot \omega_{r_3}^{j\langle s\boldsymbol{u}'_{\boldsymbol{w}} + \boldsymbol{w}, \boldsymbol{v} - \widetilde{\boldsymbol{v}} \rangle / r_2} \right) \\
&= \left(\sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} p_{\boldsymbol{w}}^2 \right) \cdot \left(\sum_{\boldsymbol{v}, \widetilde{\boldsymbol{v}} \in V} \frac{1}{t^2} \cdot \sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} \omega_{r_3}^{j\langle s\boldsymbol{u}'_{\boldsymbol{w}} + \boldsymbol{w}, \boldsymbol{v}^{[r_2]} - \widetilde{\boldsymbol{v}}^{[r_2]} \rangle} \right) \\
&= \left(\sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} p_{\boldsymbol{w}}^2 \right) \cdot \left(\sum_{\boldsymbol{v}, \widetilde{\boldsymbol{v}} \in V} \frac{1}{t^2} \cdot \sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} \omega_{r_3}^{j\langle \boldsymbol{w}, \boldsymbol{v}^{[r_2]} - \widetilde{\boldsymbol{v}}^{[r_2]} \rangle} \right) \\
&= \left(\sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} p_{\boldsymbol{w}}^2 \right) \cdot \left(\sum_{\boldsymbol{v}, \widetilde{\boldsymbol{v}} \in V} \frac{1}{t^2} \cdot s^n \cdot 1_{\boldsymbol{v}^{[r_2]} \neq \widetilde{\boldsymbol{v}}^{[r_2]}} \right) \\
&= \left(\sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} p_{\boldsymbol{w}}^2 \right) \cdot \left(\sum_{\boldsymbol{w} \in \mathbb{Z}_s^n} q_{\boldsymbol{w}}^2 \right) \cdot s^n.
\end{aligned} \tag{7}$$

In the last step we used the fact $\boldsymbol{v}^{[r_2]} \neq \widetilde{\boldsymbol{v}}^{[r_2]}$ for two $\boldsymbol{v}, \widetilde{\boldsymbol{v}} \in V$. This can be seen by contrapositive. If $\boldsymbol{v}^{[r_2]} = \widetilde{\boldsymbol{v}}^{[r_2]}$, we have $\boldsymbol{v} = \widetilde{\boldsymbol{v}}$ since $\boldsymbol{v}^{(r_2)} = \widetilde{\boldsymbol{v}}^{(r_2)}$. This contradicts Claim 2.5.

By (7), we can see that either $\sum p_{\boldsymbol{w}}^2 \geq 1/(s^{n/2+2} f(s)^2)$ or $\sum q_{\boldsymbol{w}}^2 \geq 1/(s^{n/2+2} f(s)^2)$ holds. Without loss of generality, we assume $\sum p_{\boldsymbol{w}}^2 \geq 1/(s^{n/2+2} f(s)^2)$. By

$$\max\{p_{\boldsymbol{w}}\} = \max\{p_{\boldsymbol{w}}\} \cdot \sum p_{\boldsymbol{w}} \geq \sum p_{\boldsymbol{w}}^2,$$

there exists a bucket $\widetilde{B}(\boldsymbol{w}_0, U)$ with size at least $t/(s^{n/2+2} f(s)^2)$. Let $\widetilde{U}$ be that bucket, and $\widetilde{V}$ be the subset of V with the same indices. Then $(\widetilde{U}, \widetilde{V}) \subseteq (U, V)$

is an MV family of size at least $t/(s^{n/2+2}f(s)^2)$. Next, we will find a subfamily $(U', V') \subseteq (\widetilde{U}, \widetilde{V})$ that respects $(r_1 s, r_2, r_3/s)$.

Step 4: Analyzing the elements in the large bucket. Let $\boldsymbol{u}_0$ and $\boldsymbol{v}_0$ denote $\boldsymbol{u}^{(r_1)}$ and $\boldsymbol{v}^{(r_2)}$ respectively for $\boldsymbol{u} \in U, \boldsymbol{v} \in V$. For every $\boldsymbol{u} \in \widetilde{U}$, we know $(\boldsymbol{u}^{[r_1]} - \widetilde{\boldsymbol{u}}^{[r_1]})^{(s)}$ equals the same vector $\boldsymbol{w}_0$ by the definition of the bucket. Therefore $\boldsymbol{u}^{[r_1]} - \widetilde{\boldsymbol{u}}^{[r_1]} = s\boldsymbol{u}' + \boldsymbol{w}_0$ and

$$\begin{aligned} \boldsymbol{u} &= r_1 \boldsymbol{u}^{[r_1]} + \boldsymbol{u}_0 = r_1(\widetilde{\boldsymbol{u}}^{[r_1]} + s\boldsymbol{u}' + \boldsymbol{w}_0) + \boldsymbol{u}_0 \\ &= r_1 s\boldsymbol{u}' + \left(r_1 \widetilde{\boldsymbol{u}}^{[r_1]} + r_1 \boldsymbol{w}_0 + \boldsymbol{u}_0\right). \end{aligned} \tag{8}$$

We can see $\boldsymbol{u}^{(r_1 s)} = r_1 \widetilde{\boldsymbol{u}}^{[r_1]} + r_1 \boldsymbol{w}_0 + \boldsymbol{u}_0$ is the same for all $\boldsymbol{u} \in \widetilde{U}$. Also $\boldsymbol{v}^{(r_2)} = \boldsymbol{v}_0$ is the same for all $\boldsymbol{v} \in \widetilde{V}$. These two conditions are still satisfied for any subfamily of $(\widetilde{U}, \widetilde{V})$. It suffices to find $(U', V') \subseteq (\widetilde{U}, \widetilde{V})$ such that

- $\langle \boldsymbol{u}^{[r_1 s]}, \boldsymbol{v}_0 \rangle$ modulo r_2 is the same for all $\boldsymbol{u} \in U'$. By (8) we have $\boldsymbol{u}^{[r_1 s]} = \boldsymbol{u}'$, so we need $\langle \boldsymbol{u}', \boldsymbol{v}_0 \rangle$ modulo r_2 to be the same for all $\boldsymbol{u} \in U'$.
- $\langle r_1 \widetilde{\boldsymbol{u}}^{[r_1]} + r_1 \boldsymbol{w} + \boldsymbol{u}_0, \boldsymbol{v}^{[r_2]} \rangle$ modulo $r_1 s$ is the same for all $\boldsymbol{v} \in V'$.

Since $\langle \boldsymbol{u}^{[r_1]}, \boldsymbol{v}_0 \rangle = \langle s\boldsymbol{u}' + \widetilde{\boldsymbol{u}}^{[r_1]} + \boldsymbol{w}, \boldsymbol{v}_0 \rangle$ modulo r_2 is the same for all $\boldsymbol{u} \in U$ by (U, V) respecting (r_1, r_2, r_3), we can see that $s\langle \boldsymbol{u}', \boldsymbol{v}_0 \rangle$ modulo r_2 is the same for all $\boldsymbol{u} \in U$. Hence there are $\gcd(s, r_2)$ possible values for $\langle \boldsymbol{u}', \boldsymbol{v}_0 \rangle$ modulo r_2. We pick the most frequent value c_1 and keep only the vectors with $\langle \boldsymbol{u}', \boldsymbol{v}_0 \rangle \equiv c_1 \pmod{r_2}$ in $\widetilde{U}$ and the corresponding vectors in $\widetilde{V}$.

Since $\langle \boldsymbol{u}_0, \boldsymbol{v}^{[r_2]} \rangle$ modulo r_1 is the same for all $\boldsymbol{v} \in V$ by (U, V) respecting (r_1, r_2, r_3), we can see that there are s possible values for $\langle \boldsymbol{u}_0, \boldsymbol{v}^{[r_2]} \rangle$ modulo sr_1. We pick the most frequent value c_2 and keep only the vectors with $\langle \boldsymbol{u}_0, \boldsymbol{v}^{[r_2]} \rangle \equiv c_2 \pmod{sr_1}$ in $\widetilde{U}$ and the corresponding vectors in $\widetilde{V}$.

After the above two steps, the MV family has size at least

$$\frac{|(\widetilde{U}, \widetilde{V})|}{\gcd(s, r_2) \cdot s} \geq \frac{|(\widetilde{U}, \widetilde{V})|}{s^2} \geq \frac{t}{s^{n/2+4} f(s)^2}. \tag{9}$$

And this is the required (U', V'). □

4 Proof of Theorems 1.1 and 1.2

We now prove Theorem 1.1 by repeatedly applying Lemma 3.1.

Proof (Proof of Theorem 1.1). By Claim 2.10, (U, V) is good with respect to $(1, 1, m)$. Initially we set $r_1 = 1$, $r_2 = 1$ and $r_3 = m$. By Lemma 3.1, we there is a subfamily that respects (r_1', r_2', r_3'), where $r_1' r_2' r_3' = m$ and $r_3' < m$. We repeatedly apply Lemma 3.1. Each round r_3 is reduced by some factor. We can continue this procedure until either $r_3 = 1$ or the size of the MV family becomes less than $100m$. For the case $r_3 = 1$, the size of the MV family is also less

than $100m$ by Claim 2.11. Say there are k rounds, and in each round we divide r_3 by $s_1, s_2, \ldots, s_k$ respectively. We have $s_1 s_2 \cdots s_k \leq m$ and in the ith round ($i \in [k]$), the size of the MV family is decreased by a factor at most $s_i^{n/2+4} f(s_i)^2$. Therefore the original size is upper bounded by

$$|(U,V)| \leq 100m \cdot \prod_{i=1}^{k} s_i^{n/2+4} f(s_i)^2 \leq 100m \cdot m^{n/2+4} \cdot \prod_{i=1}^{k} f(s_i)^2$$
$$= 100m^{n/2+5} \prod_{i=1}^{k} f(s_i)^2.$$

Pick $f(s) = s^{1.735}$, we can verify that $\sum_{s=2}^{\infty} \frac{1}{f(s)} \leq 0.99$. Therefore $|(U,V)| \leq 100m^{n/2+5}(m^{1.735})^2 = 100m^{n/2+8.47}$. □

Combining with the lower bound $m^{n-1+o_m(1)}$ proved in [DGY11], we can give a universal lower bound for the length of the MV code in [DGY11]. This is a restatement of Theorem 1.2 stated in the introduction.

Corollary 4.1. *Any MV code (as constructed in [DGY11]) has encoding length at least $N > K^{\frac{19}{18}}$, where K is the message length regardless of the query complexity.*

Proof. Given an MV family in $\mathbb{Z}_m^n$ with size t, we can encode a message of length $K = t$ into a codeword of length $N = m^n$.

If $n \geq 19$, by Theorem 1.1 we have $K \leq m^{n/2+8.47}$. Hence $K \leq m^{(1/2+8.47/19)n} < m^{\frac{18}{19}n} = N^{\frac{18}{19}}$ and $N > K^{\frac{19}{18}}$.

If $n \leq 18$, it was shown in [DGY11] that $K \leq m^{n-1+o_m(1)}$. Hence $K < m^{n-\frac{18}{19}} \leq m^{n-\frac{n}{19}} = m^{\frac{18}{19}n} = N^{\frac{18}{19}}$ and $N > K^{\frac{19}{18}}$. Note that here we assumed m is sufficient large. This is reasonable because we are considering encoding an arbitrarily long message and K is sufficiently large. □

5 The Case of Distinct Prime Factors

If m is a product of distinct primes, the bound can be improved to $m^{n/2+4+o_m(1)}$. The proof follows the same outline as general composite m.

Theorem 5.1. *Let m be a product of distinct primes. For every MV family (U,V) in $\mathbb{Z}_m^n$, $|(U,V)| \leq 100m^{n/2+4+o_m(1)}$, where $o_m(1)$ goes to 0 as m grows.*

Proof. The proof is similar to Theorem 1.1. We only sketch the changes here.

First, we improve the size of the (U', V') found in Lemma 3.1 to $t/(s^{n/2+2} f(s)^2)$. Since m is a product of distinct primes, r_1 and r_2 must be coprime to s, where s is the number in inequality (3). Let τ_1 and τ_2 be integers that $\tau_1 r_1 \equiv 1 \pmod{s}$ and $\tau_2 r_2 \equiv 1 \pmod{s}$, we have

$$\omega_{r_3}^{j\frac{\langle \boldsymbol{u},\boldsymbol{v}\rangle}{r_1 r_2}} = \omega_{r_3}^{j\langle \boldsymbol{u},\boldsymbol{v}\rangle \tau_1 \tau_2}.$$

We partition U and V into buckets according to $\boldsymbol{u}$ modulo s and $\boldsymbol{v}$ modulo s: $U = \bigcup_{\boldsymbol{w}\in\mathbb{Z}_s^n} B(\boldsymbol{w}, U)$ and $V = \bigcup_{\boldsymbol{w}\in\mathbb{Z}_s^n} B(\boldsymbol{w}, V)$, where

$$B(\boldsymbol{w}, U) = \{\boldsymbol{u} \in U \mid \boldsymbol{u}^{(s)} = \boldsymbol{w}\}$$

and

$$B(\boldsymbol{w}, V) = \{\boldsymbol{v} \in V \mid \boldsymbol{v}^{(s)} = \boldsymbol{w}\}.$$

We still use $p_{\boldsymbol{w}}$ to denote $|B(\boldsymbol{w}, U)|/t$ and $q_{\boldsymbol{w}}$ to denote $|B(\boldsymbol{w}, V)|/t$. By inequality (3),

$$\begin{aligned}
\left(\frac{1}{sf(s)}\right)^2 &\le \left| \mathop{\mathbb{E}}_{\substack{\boldsymbol{u}\sim U\\ \boldsymbol{v}\sim V}} \left[\omega_{r_3}^{j\langle \boldsymbol{u},\boldsymbol{v}\rangle \tau_1\tau_2}\right] \right|^2 \\
&= \left| \sum_{\boldsymbol{w}\in\mathbb{Z}_s^n} \sum_{\boldsymbol{v}\in V} p_{\boldsymbol{w}} \cdot \frac{1}{t} \cdot \omega_{r_3}^{j\langle \boldsymbol{w},\boldsymbol{v}\rangle \tau_1\tau_2} \right|^2 \\
&\le \left(\sum_{\boldsymbol{w}\in\mathbb{Z}_s^n} p_{\boldsymbol{w}}^2 \right) \cdot \left(\sum_{\boldsymbol{w}\in\mathbb{Z}_s^n} \left| \sum_{\boldsymbol{v}\in V} \frac{1}{t} \cdot \omega_{r_3}^{j\langle \boldsymbol{w},\boldsymbol{v}\rangle \tau_1\tau_2} \right|^2 \right) \\
&= \left(\sum_{\boldsymbol{w}\in\mathbb{Z}_s^n} p_{\boldsymbol{w}}^2 \right) \cdot \left(\sum_{\boldsymbol{v},\widetilde{\boldsymbol{v}}\in V} \frac{1}{t^2} \cdot \sum_{\boldsymbol{w}\in\mathbb{Z}_s^n} \omega_{r_3}^{j\langle \boldsymbol{w},\boldsymbol{v}-\widetilde{\boldsymbol{v}}\rangle \tau_1\tau_2} \right) \\
&= \left(\sum_{\boldsymbol{w}\in\mathbb{Z}_s^n} p_{\boldsymbol{w}}^2 \right) \cdot \left(\sum_{\boldsymbol{w}\in\mathbb{Z}_s^n} q_{\boldsymbol{w}}^2 \right) \cdot s^n.
\end{aligned}$$

We can see either $\sum p_{\boldsymbol{w}}^2 \ge 1/(s^{n/2+1} f(s))$ or $\sum q_{\boldsymbol{w}}^2 \ge 1/(s^{n/2+1} f(s))$. Without loss of generality, assume $\sum p_{\boldsymbol{w}}^2 \ge 1/(s^{n/2+1} f(s))$. By

$$\max\{p_{\boldsymbol{w}}\} = \max\{p_{\boldsymbol{w}}\} \cdot \sum p_{\boldsymbol{w}} \ge \sum p_{\boldsymbol{w}}^2,$$

there exists a bucket with size $|B(\boldsymbol{w}, U)| \ge t/(s^{n/2+1} f(s))$. Let $\widetilde{U}$ be that bucket, and $\widetilde{V}$ be the subset of V with the same indices. Then $(\widetilde{U}, \widetilde{V}) \subseteq (U, V)$ is an MV family of size at least $t/(s^{n/2+1} f(s))$.

Then we can find $(U', V') \subseteq (\widetilde{U}, \widetilde{V})$ using the same method as in Lemma 3.1. By inequality (9),

$$|(U', V')| \ge \frac{|(\widetilde{U}, \widetilde{V})|}{\gcd(s, r_2) \cdot s} = \frac{|(\widetilde{U}, \widetilde{V})|}{s} \ge \frac{t}{s^{n/2+2} f(s)}.$$

At last, we use the proof of Theorem 1.1 except $f(s) = 3s \ln^2 s$. One can verify $\sum_{s=2}^{\infty} \frac{1}{f(s)} < 0.99$. Let $s_1, s_2, \ldots, s_k$ be the numbers divided from r_3 in each round, by the proof of Theorem 1.1,

$$|(U,V)| \leq 100m \prod_{i=1}^{k} s_i^{n/2+2} f(s_i) \leq 100m^{n/2+3} \prod_{i=1}^{k} (3s_i \ln^2 s_i)$$
$$\leq 100m^{n/2+4} \prod_{i=1}^{k} (3 \ln^2 s_i).$$

For a sufficiently large integer s, we have $3 \ln^2 s < s^{\epsilon}$, where ϵ is an arbitrary fixed small number. When $m \to \infty$, all $s_1, s_2, \ldots, s_k$ except a constant number of them must be that large. Take $\epsilon \to 0$, we have $|(U,V)| \leq m^{n/2+4+o_m(1)}$. □

References

[BBR94] Barrington, D.A.M., Beigel, R., Rudich, S.: Representing boolean functions as polynomials modulo composite numbers. In: Computational Complexity, pp. 455–461 (1994)

[BDL13] Bhowmick, A., Dvir, Z., Lovett, S.: New bounds on matching vector families. In: 45th ACM Symposium on Theory of Computing, STOC (2013)

[BET10] Ben-Aroya, A., Efremenko, K., Ta-Shma, A.: Local list decoding with a constant number of queries. In: 51st IEEE Symposium on Foundations of Computer Science (FOCS), pp. 715–722 (2010)

[BF98] Babai, L., Frankl, P.: Linear algebra methods in combinatorics (1998)

[CFL+10] Chee, Y.M., Feng, T., Ling, S., Wang, H., Zhang, L.F.: Query-efficient locally decodable codes of subexponential length. Electronic Colloquium on Computational Complexity (ECCC), TR10-173 (2010)

[DGY11] Dvir, Z., Gopalan, P., Yekhanin, S.: Matching vector codes. SIAM J. Comput. 40(4), 1154–1178 (2011)

[Efr09] Efremenko, K.: 3-query locally decodable codes of subexponential length. In: 41st ACM Symposium on Theory of Computing (STOC), pp. 39–44 (2009)

[GKST02] Goldreich, O., Karloff, H., Schulman, L.J., Trevisan, L.: Lower bounds for linear locally decodable codes and private information retrieval. In: 17th IEEE Computational Complexity Conference (CCC), pp. 175–183 (2002)

[Gro00] Grolmusz, V.: Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs. Combinatorica 20(1), 71–86 (2000)

[IS10] Itoh, T., Suzuki, Y.: Improved constructions for query-efficient locally decodable codes of subexponential length. IEICE Transactions on Information and Systems E93-D(2), 263–270 (2010)

[KdW04] Kerenidis, I., de Wolf, R.: Exponential lower bound for 2-query locally decodable codes via a quantum argument. Journal of Computer and System Sciences 69(3), 395–420 (2004)

[KT00] Katz, J., Trevisan, L.: On the efficiency of local decoding procedures for error-correcting codes. In: 32nd ACM Symposium on Theory of Computing (STOC), pp. 80–86 (2000)

[KY09] Kedlaya, K.S., Yekhanin, S.: Locally decodable codes from nice subsets of finite fields and prime factors of Mersenne numbers. SIAM J. Comput. 38(5), 1952–1969 (2009)

[Rag07] Raghavendra, P.: A note on Yekhanin's locally decodable codes. Electronic Colloquium on Computational Complexity (ECCC), TR07-016 (2007)

[TV07] Tao, T., Vu, V.H.: Additive Combinatorics (2007)

[Woo07] Woodruff, D.P.: New lower bounds for general locally decodable codes. Electronic Colloquium on Computational Complexity (ECCC), TR07-006 (2007)

[Woo10] Woodruff, D.P.: A quadratic lower bound for three-query linear locally decodable codes over any field. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, pp. 766–779. Springer, Heidelberg (2010)

[Yek08] Yekhanin, S.: Towards 3-query locally decodable codes of subexponential length. Journal of the ACM 55(1), 1–16 (2008)

[YGK12] Yuan, C., Guo, Q., Kan, H.: A novel elementary construction of matching vectors. Information Processing Letters 112(12), 494–496 (2012)

Explicit Noether Normalization for Simultaneous Conjugation via Polynomial Identity Testing

Michael A. Forbes[1,⋆] and Amir Shpilka[2,⋆⋆]

[1] MIT CSAIL,
32 Vassar St.,
Cambridge, MA 02139
miforbes@mit.edu
[2] Faculty of Computer Science,
Technion — Israel Institute of Technology,
Haifa, Israel
shpilka@cs.technion.ac.il

Abstract. Mulmuley [Mul12a] recently gave an explicit version of Noether's Normalization Lemma for ring of invariants of matrices under simultaneous conjugation, under the conjecture that there are deterministic black-box algorithms for polynomial identity testing (PIT). He argued that this gives evidence that constructing such algorithms for PIT is beyond current techniques. In this work, we show this is not the case. That is, we improve Mulmuley's reduction and correspondingly weaken the conjecture regarding PIT needed to give explicit Noether Normalization. We then observe that the weaker conjecture has recently been nearly settled by the authors ([FS12]), who gave quasipolynomial size hitting sets for the class of read-once oblivious algebraic branching programs (ROABPs). This gives the desired explicit Noether Normalization unconditionally, up to quasipolynomial factors.

As a consequence of our proof we give a deterministic parallel polynomial-time algorithm for deciding if two matrix tuples have intersecting orbit closures, under simultaneous conjugation.

Finally, we consider the depth-3 diagonal circuit model as defined by Saxena [Sax08], as PIT algorithms for this model also have implications in Mulmuley's work. Previous works (such as [ASS13] and [FS12]) have given quasipolynomial size hitting sets for this model. In this work, we give a much simpler construction of such hitting sets, using techniques of Shpilka and Volkovich [SV09].

1 Introduction

Many results in mathematics are non-constructive, in the sense that they establish that certain mathematical objects exist, but do not give an efficient or

⋆ This work supported by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370.

⋆⋆ The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 527–542, 2013.

explicit construction of such objects, and often further work is needed to find constructive arguments. Motivated by the recent results of Mulmuley [Mul12a] (henceforth "Mulmuley", but theorem and page numbering will refer to the full version [Mul12b]), this paper studies constructive versions of the Noether Normalization Lemma from commutative algebra. The lemma, as used in this paper, can be viewed as taking a commutative ring R, and finding a smaller subring $S \subseteq R$ such that S captures many of the interesting properties of R (see Section 1.2 for a formal discussion). Like many arguments in commutative algebra, the usual proof of this lemma does not focus on the computational considerations of how to find (a concise representation of) the desired subring S. However, the area of computational commutative algebra (eg, [DK02, CLO07]) offers methods showing how many classic results can be made constructive, and in certain cases, the algorithms are even efficient.

While constructive methods for Noether Normalization are known using Gröbner bases (cf. [CLO07]), the Gröbner basis algorithms are not efficient in the worst-case (as show by Mayr and Meyer [MM82]), and are not known to be more efficient for the problems we consider. Diverging from the Gröbner basis idea, Mulmuley recently observed that a constructive version of Noether Normalization is really a problem in *derandomization*. That is, given the ring R, if we take a sufficiently large set of "random" elements from R, then these elements will generate the desired subring S of R. Indeed, the usual proof of Noether Normalization makes this precise, with the appropriate algebraic meaning of "random". This view suggests that random sampling from R is sufficient to construct S, and this sampling will be efficient if R itself is explicitly given. While this process uses lots of randomness, the results of the derandomization literature in theoretical computer science (eg, [IW97, IKW02, KI04]) give strong conjectural evidence that randomness in efficient algorithms is not necessary. Applied to the problems here, there is thus strong conjectural evidence that the generators for the subring S can be constructed efficiently, implying that the Noether Normalization Lemma can be made constructive.

Motivated by this connection, Mulmuley explored what are the minimal derandomization conjectures necessary to imply an explicit form of Noether Normalization. The existing conjectures come in two flavors. Most derandomization hypotheses concern boolean computation, and as such are not well-suited for algebraic problems (for example, a single real number can encode infinitely many bits), but Mulmuley does give some connections in this regime. Other derandomization hypotheses directly concern algebraic computation, and using them Mulmuley gives an explicit Noether Normalization Lemma, for some explicit rings of particular interest. In particular, Mulmuley proves that it would suffice to derandomize the *polynomial identity testing* (PIT) problem in certain models, in order to obtain a derandomization of the Noether Normalization Lemma. Mulmuley actually views this connection as an evidence that derandomizing PIT for these models is a difficult computational task (Mulmuley, p. 3) and calls this the GCT chasm. Although Mulmuley conjectures that it could be crossed he strongly argues that this cannot be achieved with current techniques (Mulmuley, p. 3):

"*On the negative side, the results in this article say that black-box derandomization of PIT in characteristic zero would necessarily require proving, either directly or by implication, results in algebraic geometry that seem impossible to prove on the basis of the current knowledge.*"

Mulmuley supports this intuition with his Theorem 1.6, which gives an equivalence between the general PIT problem (considered difficult in complexity theory) and explicit Noether Normalization for explicit varieties (considered difficult in algebraic geometry). However, he also focuses on Noether Normalization of specific varieties, such as the ring of invariants of matrices under simultaneous conjugation. The conditional derandomization of Noether Normalization for this ring is the content of his Theorems 1.1 and 1.2. Explicit Noether Normalization for this ring is simpler as there are explicitly known generators for the ring of invariants (as given in previous work, cited in Theorem 1.3), and these generators are computationally very simple. For general explicit varieties, obtaining such explicit generators is an open problem, and even if found, the generators would not likely be as computational simple as the generators of Theorem 1.3.

However, Mulmuley has suggested that even the results in his Theorems 1.1 and 1.2 are beyond current techniques, because of their superficial similarity to the "wild" classification problem of matrix tuples under simultaneous conjugation (see Mulmuley, p. 5, and the public presentation [Mul12c]). Indeed, the general Noether Normalization problem for explicit varieties was not discussed in the public presentation, as the specific case of simultaneous conjugation was itself implied to contain the "seemingly impossible" problems mentioned above.

In this work, we obtain (via existing techniques) an unconditional derandomization of Noether's Normalization Lemma for simultaneous conjugation, as conditionally established in Mulmuley's Theorems 1.1 and 1.2. While we have no results for explicit Noether Normalization of general explicit varieties, we believe that our work suggests that problems cannot be assumed to be difficult just because they originated in algebraic-geometric setting, and that one has to consider the finer structure of the problem. In particular, our work suggests that the similarity of the problem we consider to the above wild problem is indeed superficial. We start by briefly describing the PIT problem. For more details, see the survey by Shpilka and Yehudayoff [SY10].

1.1 Polynomial Identity Testing

The PIT problem asks to decide whether a polynomial, given by an algebraic circuit, is the zero polynomial. An algebraic circuit is a directed acyclic graph with a single sink (or root) node, called the *output*. Internal nodes are labeled with either a $\times$- or $+$-gate, which denote multiplication and addition respectively. The source (or leaf) nodes, are labeled with either variables x_i, or elements from a given field $\mathbb{F}$. An algebraic circuit computes a polynomial in the ring $\mathbb{F}[x_1, \ldots, x_n]$ in the natural way (as there are no cycles): each internal node in the graph computes a function of its children (either $\times$ or $+$), and the circuit itself outputs the polynomial computed by the output node. Algebraic circuits give the most natural and succinct way to represent a polynomial.

Given an algebraic circuit C, the PIT problem is to test if the polynomial f it computes is identically zero, as a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$. Schwartz [Sch80] and Zippel [Zip79] (along with the weaker result by DeMillo and Lipton [DL78]) showed that if $f \not\equiv 0$ is a polynomial of degree $\leq d$, and $\alpha_1, \ldots, \alpha_n \in S \subseteq \mathbb{F}$ are chosen uniformly at random, then $f(\alpha_1, \ldots, \alpha_n) = 0$ with probability $\leq d/|S|$. It follows then that we can solve PIT efficiently using randomness, as evaluating an algebraic circuit can be done efficiently. The main question concerning the PIT problem is whether it admits an efficient *deterministic* algorithm, in the sense that it runs in polynomial time in the size of the circuit C. Heuristically, this problem can be viewed as replacing any usage of the Schwartz-Zippel result with a deterministic set of evaluations.

One important feature of the above randomized PIT algorithm is that it only uses the circuit C by evaluating it on given inputs. Such algorithms are called *black-box* algorithms, as they treat the circuit as merely a black-box that computes some low-degree polynomial (that admits some small circuit computing it). This is in contrast to *white-box* algorithms, that probe the structure of C in some way. White-box algorithms are thus less restricted, whence deriving a black-box algorithm is a stronger result. For the purposes of this paper, instead of referring to deterministic black-box PIT algorithms, we will use the equivalent notion of a *hitting set*, which is a small set $\mathcal{H} \subseteq \mathbb{F}^n$ of evaluation points such that for any non-zero polynomial f computed by a small algebraic circuit, f must evaluate to non-zero on some point in $\mathcal{H}$. A standard argument (see [SY10]) shows that small hitting sets *exist*, the main question is whether small *explicit* hitting sets, which we now define, exist. As usual, the notion of explicit must be defined with respect to an infinite family of objects, one object for each value of n. For clarity, we abuse notation and do not discuss families, with the understanding that any objects we design will belong to an unspecified family, and that there is a single (uniform) algorithm to construct these objects that takes as input the relevant parameters.

Definition 1.1. *Let $\mathcal{C} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be a class of polynomials. A set $\mathcal{H} \subseteq \mathbb{F}^n$ is a* ***hitting set for*** *$\mathcal{C}$ if for all $f \in \mathcal{C}$, $f \equiv 0$ iff $f|_{\mathcal{H}} \equiv 0$. The hitting set $\mathcal{H}$ is $t(n)$-**explicit** if there is an algorithm such that given an index into $\mathcal{H}$, the corresponding element of $\mathcal{H}$ can be computed in $t(n)$-time, assuming unit cost arithmetic in $\mathbb{F}$.*

That is, we mean that the algorithm can perform field operations (add, subtract, multiply, divide, zero test) in $\mathbb{F}$ in unit time, and can start with the constants 0 and 1. We will also assume the algorithm has access to an arbitrary enumeration of $\mathbb{F}$. In particular, when $\mathbb{F}$ has characteristic 0, without loss of generality the algorithm will only produce rational numbers.

1.2 Noether Normalization for Simultaneous Conjugation

Mulmuley showed that when R is a particular ring, then the problem of finding the subring S given by Noether Normalization can be reduced to the black-box PIT problem, so that explicit hitting sets (of small size) would imply a

constructive version of Noether Normalization for this ring. The ring considered here and in Mulmuley's Theorems 1.1 and 1.2 is the ring of invariants of matrices, under the action of simultaneous conjugation[1].

Definition 1.2. *Let $\vec{M}$ denote a vector of r matrices, each*[2] $[\![n]\!] \times [\![n]\!]$*, whose entries are distinct variables. Consider the action of* $\mathrm{GL}_n(\mathbb{F})$ *by simultaneous conjugation on $\vec{M}$, that is, $M_1, \ldots, M_r) \mapsto (PM_1P^{-1}, \ldots, PM_rP^{-1})$. Define $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ to be the subring of $\mathbb{F}[\vec{M}]$ consisting of polynomials in the entries of $\vec{M}$ that are invariant under the action of* $\mathrm{GL}_n(\mathbb{F})$*. That is, $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})} := \{f | f(\vec{M}) = f(P\vec{M}P^{-1}), \forall P \in \mathrm{GL}_n(\mathbb{F})\}$.*

Note that $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ is in fact a ring. When $\mathbb{F}$ has characteristic zero, the following result gives an explicit set of generators for the ring of invariants. When $\mathbb{F}$ has positive characteristic, the result is known not to hold (see [KP00, §2.5]) so we will only discuss characteristic zero fields.

Theorem 1.3 ([Pro76, Raz74, For86]). *Let $\mathbb{F}$ be a field of characteristic zero. Let $\vec{M}$ denote a vector of r matrices, each $[\![n]\!] \times [\![n]\!]$, whose entries are distinct variables. The ring $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ of invariants is generated by the invariants $\mathcal{T} := \{\mathrm{trace}(M_{i_1} \cdots M_{i_\ell}) | \vec{i} \in [\![r]\!]^\ell, \ell \in [n^2]\}$.*

Further, the ring $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ is not generated by the invariants $\{\mathrm{trace}(M_{i_1} \cdots M_{i_\ell}) | \vec{i} \in [\![r]\!]^\ell, \ell \in [\lceil n^2/8 \rceil]\}$.

That is, every invariant can be represented as a (multivariate) polynomial, with coefficients in $\mathbb{F}$, in the above generating set. Note that the above generating set is indeed a set of invariants, because the trace is cyclic, so the action of simultaneous conjugation by P cancels out.

The above result is explicit in two senses. The first sense is that all involved field constants can be efficiently constructed. The second is that for any $f \in \mathcal{T}$ and $\vec{A}$, $f(\vec{A})$ can be computed quickly. In particular, any $f \in \mathcal{T}$ can be computed by a $\mathsf{poly}(n, r)$-sized algebraic circuit, as matrix multiplication and trace can be computed efficiently by circuits. We encapsulate these notions in the following definition.

Definition 1.4. *A set $\mathcal{P} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ of polynomials has $t(n)$-**explicit $\mathcal{C}$-circuits**, if there is an algorithm such that given an index into $\mathcal{P}$, a circuit $C \in \mathcal{C}$ can be computed in $t(n)$-time, assuming unit cost arithmetic in $\mathbb{F}$, such that C computes the indexed $f \in \mathcal{P}$.*

In particular, the above definition implies that the resulting circuits C have size at most $t(n)$. The class of circuits $\mathcal{C}$ can be the class of all algebraic circuits, or

[1] Mulmuley considers conjugation by matrices with determinant 1, that is, by SL_n. Over algebraically closed fields, which are the main fields of interest in this paper, this is equivalent to conjugation over GL_n.

[2] In this work we will most often index vectors and matrices starting at zero, and will indicate this by the use of $[\![n]\!]$, which denotes the set $\{0, \ldots, n-1\}$. Also, $[n]$ will be used to denote the set $\{1, \ldots, n\}$.

some restricted notion, such as algebraic branching programs, which are defined later in this paper. Thus, in the language of the above definition, the set of generators $\mathcal{T}$ has $\mathsf{poly}(n, r)$-explicit algebraic circuits.

However, the above result is unsatisfactory in that the set of generators $\mathcal{T}$ has size $\exp(\mathsf{poly}(n, r))$, which is unwieldy from a computational perspective. One could hope to find a smaller subset of generators, but the lower bound in the above theorem rules out one approach that direction. The number of generators is relevant here, as we will consider three computational problems where these generators are useful, but because of their number the resulting algorithms will be exponential-time, where one could hope for something faster. To define these problems, we first give the following standard definition from commutative algebra.

Definition 1.5. *Let R be a commutative ring, and S a subring. Then R is **integral** over S if every element in R satisfies some monic polynomial with coefficients in S.*

As an example, the algebraic closure of $\mathbb{Q}$ (the algebraic numbers) is integral over $\mathbb{Q}$. In this work the rings R and S will be rings of polynomials, and it is not hard to see that all polynomials in R vanish at a point iff all polynomials in S vanish at that point. This can be quite useful, especially if S has a small list of generators. The statement of Noether Normalization is exactly that of providing such an S with a small list of generators. The question we consider here is how to find an explicit such S for the ring of invariants under simultaneous conjugation, where S should be given by its generators.

Question 1.6 (Constructive Noether Normalization). Let $\mathbb{F}$ be an algebraically closed field of characteristic zero. Is there a small set of polynomials $\mathcal{T}' \subseteq \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ with explicit algebraic circuits, such that $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ is integral over the subring S generated by $\mathcal{T}'$?

We will in fact mostly concern ourselves with the next problem, which has implications for the first, when $\mathbb{F}$ is algebraically closed. We first give the following definition, following Derksen and Kemper [DK02].

Definition 1.7. *A subset $\mathcal{T}' \subseteq \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ is a set of **separating invariants** if for all $\vec{A}, \vec{B} \in (\mathbb{F}^{[n]\times[n]})^{[r]}$ there exists an $f \in \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ such that $f(\vec{A}) \neq f(\vec{B})$ iff there exists an $f' \in \mathcal{T}'$ such that $f'(\vec{A}) \neq f'(\vec{B})$.*

In words, whenever there is an $f \in \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ separating $\vec{A}$ and $\vec{B}$ (i.e. $f(\vec{A}) \neq f(\vec{B})$), there is also an $f' \in \mathcal{T}'$ separating them. As before, we will ask whether we can find an explicit construction.

Question 1.8. Let $\mathbb{F}$ have characteristic zero. Is there a small set of separating invariants $\mathcal{T}' \subseteq \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ with explicit algebraic circuits?

Mulmuley used the tools of geometric invariant theory [MFK94], as done in Derksen and Kemper [DK02], to note that, over algebraically closed fields, any

set $\mathcal{T}'$ of separating invariants will also generate a subring that $(\mathbb{F}^{[\![n]\!]\times[\![n]\!]})^{[\![r]\!]}$ is integral over. Thus, any positive answer to Question 1.8 will give a positive answer to Question 1.6. Hence, we will focus on constructing explicit separating invariants (over any field of characteristic zero).

Note that relaxations of Question 1.8 can be answered positively. If we only insist on explicit separating invariants (relaxing the insistence on having few invariants), then the exponentially-large set of generators $\mathcal{T}$ given in Theorem 1.3 suffices as these polynomials have small circuits and as they generate the ring of invariants, they have the required separation property. In contrast, if we only insist of a small set of separating invariants (relaxing the explicitness), then Noether Normalization essentially shows that a *non-explicit* set of separating invariants $\mathcal{T}'$ of size $\mathsf{poly}(n, r)$ exists, basically by taking a random $\mathcal{T}'$. More constructively, Mulmuley observed that Gröbner basis techniques can construct a small set of separating invariants $\mathcal{T}'$, but this set is still not explicit as such algorithms take exponential-space, so are far from efficient. In the particular case of $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$, Mulmuley showed that the construction can occur in PSPACE unconditionally, or even PH, assuming the Generalized Riemann Hypothesis. Thus, while there are explicit sets of separating invariants, and there are small sets of separating invariants, existing results do not achieve these two properties simultaneously.

The third problem is more geometric, as opposed to algebraic. Given a tuple of matrices $\vec{A}$, we can consider the orbit of $\vec{A}$ under simultaneous conjugation as a subset of $(\mathbb{F}^{[\![n]\!]\times[\![n]\!]})^{[\![r]\!]}$. A natural computational question is to decide whether the orbits of $\vec{A}$ and $\vec{B}$ intersect. However, from the perspective of algebraic geometry it is more natural to ask of the *orbit closures* intersect. That is, we now consider $\vec{A}$ and $\vec{B}$ as lying in $(\overline{\mathbb{F}}^{[\![n]\!]\times[\![n]\!]})^{[\![r]\!]}$, where $\overline{\mathbb{F}}$ is the algebraic closure of $\mathbb{F}$. Then, we consider the orbit closures of $\vec{A}$ and $\vec{B}$ in this larger space, where this refers to taking the orbits in $(\overline{\mathbb{F}}^{[\![n]\!]\times[\![n]\!]})^{[\![r]\!]}$ and closing them with respect to the Zariski topology.

Question 1.9. Let $\mathbb{F}$ be a field of characteristic zero. Is there an efficient deterministic algorithm (in the unit cost arithmetic model) that, given $\vec{A}, \vec{B} \in (\mathbb{F}^{[\![n]\!]\times[\![n]\!]})^{[\![r]\!]}$, decides whether the orbit closures of $\vec{A}$ and $\vec{B}$ under simultaneous conjugation have an empty intersection?

Mulmuley observed that by the dictionary of geometric invariant theory [MFK94], $\vec{A}$ and $\vec{B}$ have a non-empty intersection of their orbit closures iff they are not distinguishable by any set of separating invariants. Thus, any explicit set $\mathcal{T}'$ of separating invariants, would answer this question, as one could test if f agrees on $\vec{A}$ and $\vec{B}$ (as f is easy to compute, as it has a small circuit), for all $f \in T'$. Thus, as before, Question 1.9 can be solved positively by a positive answer to Question 1.8. The main results of this paper provide positive answers to Questions 1.6, 1.8 and 1.9.

1.3 Our Results

We study further the connection raised by Mulmuley regarding the construction of separating invariants and the black-box PIT problem. In particular, we more carefully study the classes of algebraic circuits arising in the reduction from Noether Normalization to PIT. Two models are particularly important, and we define them now.

Definition 1.10 (Nisan [Nis91]). *A **algebraic branching program with unrestricted weights** of depth d and width $\leq w$, on the variables $x_1, \ldots, x_n$, is a directed acyclic graph such that 1. The vertices are partitioned in $d+1$ layers $V_0, \ldots, V_d$, so that $V_0 = \{s\}$ (s is the source node), and $V_d = \{t\}$ (t is the sink node). Further, each edge goes from V_{i-1} to V_i for some $0 < i \leq d$. 2. $\max |V_i| \leq w$. 3. Each edge e is weighted with a polynomial $f_e \in \mathbb{F}[\vec{x}]$.*

Each s-t path is said to compute the polynomial which is the product of the labels of its edges, and the algebraic branching program itself computes the sum over all s-t paths of such polynomials.

*1. In an **algebraic branching program (ABP)**, for each edge e the weight $f_e(\vec{x})$ is an affine function. The **size** is nwd. 2. In a **read-once oblivious ABP (ROABP)** of (individual) degree $< r$, we have $n := d$, and for each edge e from V_{i-1} to V_i, the weight is a univariate polynomial $f_e(x_i) \in \mathbb{F}[x_i]$ of degree $< r$. The **size** is dwr.*

In the definition of ROABPs we will exclusively focus on individual degree, and thus will use the term "degree". The ROABP model is called *oblivious* because the variable order $x_1 < \cdots < x_d$ is fixed. The model is called *read-once* because the variables are only accessed on one layer in the graph.

The ABP model is a standard algebraic model that is at least as powerful as algebraic formulas, as shown by Valiant [Val79], and can be simulated by algebraic circuits. As shown by Berkowitz [Ber84], the determinant can be computed by a small ABP over any field. See Shpilka and Yehuydayoff [SY10] for more.

The ROABP model arose in prior work of the authors ([FS12]) as a natural model of algebraic computation capturing several other existing models. This model can also be seen as an algebraic analogue of the boolean model of computation known as the read-once oblivious branching program model, which is a non-uniform analogue of the complexity class RL. See Forbes and Shpilka [FS12] for more of a discussion on the motivation of this class.

A polynomial computed by an ROABP of size s can be computed by an ABP of size $\mathsf{poly}(s)$. The converse is false, as Nisan [Nis91] gave exponential lower bounds for the size of non-commutative ABPs computing the determinant, and non-commutative ABPs encompass ROABPs, while as mentioned above Berkowitz [Ber84] showed the determinant can be computed by small ABPs. Thus the ROABPs are strictly weaker in computational power than ABPs.

While there are no efficient (white-box or black-box) PIT algorithms for ABPs, we established in prior work ([FS12]) a quasi-polynomial sized hitting set for ROABPs. This hitting set will be at the heart of our main result.

Theorem 1.11 ([FS12]). *Let $\mathcal{C}$ be the set of d-variate polynomials computable by depth d, width $\leq w$, degree $< r$ ROABPs. If $|\mathbb{F}| \geq \mathsf{poly}(d, w, r)$, then $\mathcal{C}$ has a $\mathsf{poly}(d, w, r)$-explicit hitting set $\mathcal{H} \subseteq \mathbb{F}^d$, of size $\leq \mathsf{poly}(d, w, r)^{\mathcal{O}(\lg d)}$. Further, if $\mathbb{F}$ has characteristic zero then $\mathcal{H} \subseteq \mathbb{Q}^d$.*

Our main contribution concerns the derandomization of Noether Normalization Lemma for the ring of matrix invariants.

Derandomizing Noether Normalization via an improved reduction to PIT: This section contains the main results of our paper. Given the above result, and the lack of progress on derandomizing PIT in such general models such as ABPs, it might seem that derandomizing Noether Normalization for simultaneous conjugation is challenging. However, we show this is not true, by showing that derandomization of black-box PIT for ROABPs suffices for derandomizing Noether Normalization for simultaneous conjugation. By then invoking our prior work on hitting sets for ROABPs cited as Theorem 1.11, we establish the following theorems, giving quasi-affirmative answers to Questions 1.6, 1.8 and 1.9. Furthermore, our results are proved unconditionally and are at least as strong as the conditional results Mulmuley obtains by assuming strong conjectures such as the Generalized Riemann Hypothesis or strong lower bound results.

Specifically, we prove the following theorem which gives an explicit set of separating invariants (see Question 1.8).

Theorem 1.12. *Let $\mathbb{F}$ be a field of characteristic zero. There is a $\mathsf{poly}(n, r)^{\mathcal{O}(\log(n))}$-sized set $\mathcal{T}_{\mathcal{H}}$ of separating invariants, with $\mathsf{poly}(n, r)$-explicit ABPs. That is, $\mathcal{T}_{\mathcal{H}} \subseteq \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$, and for any $\vec{A}, \vec{B} \in (\mathbb{F}^{[\![n]\!] \times [\![n]\!]})^{[\![r]\!]}$, $f(\vec{A}) \neq f(\vec{B})$ for some $f \in \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ iff $f'(\vec{A}) \neq f'(\vec{B})$ for some $f' \in T_{\mathcal{H}}$.*

As a consequence of Theorem 1.12 and the discussion in Subsection 1.2 we obtain the following corollary that gives a positive answer to Question 1.6. In particular, it provides a derandomization of Noether Normalization Lemma for the ring of invariants of simultaneous conjugation.

Corollary 1.13. *Let $\mathbb{F}$ be an algebraically closed field of characteristic zero. Let $\mathcal{T}_{\mathcal{H}}$ be the set guaranteed by Theorem 1.12. Then, $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ is integral over the subring generated by $T_{\mathcal{H}}$.*

For deciding intersection of orbit closures, Question 1.9, the natural extension of Theorem 1.12, as argued in Subsection 1.2, would yield a quasi-polynomial-time algorithm for deciding orbit closure intersection. However, by replacing the black-box PIT results for ROABPs of Forbes and Shpilka [FS12] by the white-box PIT results by Raz and Shpilka [RS05] (as as well as follow-up work by Arvind, Joglekar and Srinivasan [AJS09]), we can obtain the following better algorithm for deciding orbit closure intersection, proving a strong positive answer to Question 1.9.

Theorem 1.14. *Let $\mathbb{F}$ be a field of characteristic zero. There is an algorithm, running in deterministic $\mathsf{polylog}(n, r)$-time using $\mathsf{poly}(n, r)$-processors (NC), in*

the unit cost arithmetic model, such that given $\vec{A}, \vec{B} \in (\mathbb{F}^{[n]\times[n]})^{[r]}$*, one can decide whether the orbit closures of* $\vec{A}$ *and* $\vec{B}$ *under simultaneous conjugation have an empty intersection.*

PIT for Depth-3 Diagonal Circuits: Mulmuley's Theorem 1.4 showed that Noether Normalization for representations of $\mathrm{SL}_m(\mathbb{F})$ can be reduced, when m is constant, to black-box PIT of a subclass of circuits known as *depth-3 diagonal circuits*. This class of circuits (along with a depth-4 version) was introduced in Saxena [Sax08], who gave a polynomial-time white-box PIT algorithm, via a reduction to the white-box PIT algorithm for non-commutative ABPs of Raz and Shpilka [RS05]. Saha, Saptharishi and Saxena [SSS11] (among other things) generalized these results to the depth-4 semi-diagonal model. Agrawal, Saha and Saxena [ASS13] gave (among other things) a quasipolynomial size hitting set for this model, by showing that any such circuit can be shifted so that there is a small-support monomial, which can be found via brute-force. In independent work, the present authors (in [FS12]) also established (among other things) a quasipolynomial size hitting set for this model. This was done by showing that the depth-4 semi-diagonal model is efficiently simulated by the ROABP model. Further, this was done in two ways: the first was an explicit reduction by using the duality ideas of Saxena [Sax08], and the second was to show that the diagonal model has a small space of derivatives in a certain sense, and that ROABPs can efficiently compute any polynomial with that sort of small space of derivatives. Some aspects of this model are also present in the work of Gupta-Kamath-Kayal-Saptharishi [GKKS13] showing that (arbitrary) depth-3 formulas capture, in a sense, the entire complexity of arbitrary algebraic circuits.

Here, we give a simpler proof that the depth-3 diagonal circuit model has a quasipolynomial size hitting set. This is done using the techniques of [SV09], and has some similarities with the work of Agrawal, Saha and Saxena [ASS13]. In particular, we show the entire space of derivatives is small, for depth-3 model (but not the depth-4 model). We then show that this implies such polynomials must contain a monomial of logarithmic support, which can be found via brute-force in quasipolynomial time. Unlike the work of Agrawal, Saha and Saxena [ASS13], no shifts are required for this small monomial to exist. Thus, we get the following theorem.

Theorem 1.15. *Let* $\mathbb{F}$ *be a field with size* $\geq d+1$*. Then there is a* $\mathsf{poly}(n, d, \log(s))$*-explicit hitting set of size* $\mathsf{poly}(n,d)^{\mathcal{O}(\log s)}$ *for the class of* n*-variate, degree* $\leq d$*, depth-3 diagonal circuits of size* $\leq s$*.*

Deciding (non-closed) orbit membership via PIT: Finally, we observe that deciding non-closed orbit membership easily reduces to PIT. Due to lack of space we omit this, but include it in the full version [FS13].

1.4 Notation

Given a vector of polynomials $\vec{f} \in \mathbb{F}[\vec{x}]^n$ and an exponent vector $\vec{i} \in \mathbb{N}^n$, we write $\vec{f}^{\vec{i}}$ for $f_1^{i_1} \cdots f_n^{i_n}$.

Given a polynomial $f \in \mathbb{F}[\vec{x}]$, we write $\mathfrak{C}_{\vec{x}^{\vec{i}}}(f)$ to denote the coefficient of $\vec{x}^{\vec{i}}$ in f. For a matrix $M \in \mathbb{F}[\vec{x}]^{[\![r]\!]\times[\![r]\!]}$, we write $\mathfrak{C}_{\vec{x}^{\vec{i}}}(M)$ to denote the $r \times r$ $\mathbb{F}$-matrix, with the $\mathfrak{C}_{\vec{x}^{\vec{i}}}$ operator applied to each entry. When we write "$f \in \mathbb{F}[\vec{x}][\vec{y}]$", we will treat f as a polynomial in the variables $\vec{y}$, whose coefficients are polynomials in the variables $\vec{x}$, and correspondingly will write $\mathfrak{C}_{\vec{y}^{\vec{j}}}(f)$ to extract the polynomial in $\vec{x}$ that is the coefficient of the monomial $\vec{y}^{\vec{j}}$ in f.

1.5 Organization

In Section 2 we give the necessary background on ROABPs. We prove our main results about explicit Noether Normalization in Section 3. Omitted proofs can be found in the full version [FS13] of the paper.

The rest of our results appear are deferred to the full version [FS13]. These include the hitting set for depth-3 diagonal circuits and the reduction from the orbit membership problem to PIT.

2 Properties of Algebraic Branching Programs

ABPs, as well as the subclass of ROABPs, have a tight connection with matrix products. In the full version [FS13] we derive some basic properties of this connection, as they are useful for proofs of the below results. For lack of space, we omit the statements of these straightforward results.

However, we will note here a result, first proved by Malod and Portier [MP08], that shows the equivalence of traces of matrix powers (see the full version [FS13] for a definition) and algebraic branching programs.

Theorem 2.1 ([MP08]). *Let $\mathbb{F}$ be a field. If a polynomial f is computable by a width w, depth d ABP, then for any $d' \geq d$ such that $\mathrm{char}(\mathbb{F}) \nmid d'$, f can be computed by a width wd', depth d' trace of a matrix power. In particular, $d' \in \{d, d+1\}$ suffices.*

Conversely, if a polynomial f is computable by a width w, depth d trace of matrix power, then f can also be computed by a width w^2, depth d ABP.

The trace of matrix power model was model studied by Mulmuley, and he used derandomization of PIT for this class as the basis of explicit Noether Normalization. The above shows this model is equivalent to the general model of ABP. In contrast, we show a reduction to the more restricted model of ROABP which allows us to apply the results of [FS12].

3 Reducing Noether Normalization to Read-once Oblivious Algebraic Branching Programs

In this section we construct a small set of explicit separating invariants for simultaneous conjugation. We do so by constructing a single ROABP that encodes

the entire generating set $\mathcal{T}$ for $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$, as given by Theorem 1.3. We then use hitting sets for ROABPs to efficiently extract the separating invariants from this ROABP. We begin with the construction[3] of the ROABP.

Construction 3.1. *Let $n, r, \ell \geq 1$. Let $\vec{M}$ denote a vector of r matrices, each $[\![n]\!] \times [\![n]\!]$, whose entries are distinct variables. Define $M(x) := \sum_{i \in [\![r]\!]} M_i x^i$ and, for the ℓ variables $\vec{x}$, define $f_\ell(\vec{M}, \vec{x}) := \mathrm{trace}(M(x_1) \cdots M(x_\ell))$.*

The following lemma shows that these polynomials $f_\ell(\vec{M}, \vec{x})$ can be computed by small ROABPs, when $\vec{x}$ is variable and $\vec{M}$ is constant.

Lemma 3.2. *Assume the setup of Construction 3.1. Let $\vec{A}, \vec{B} \in (\mathbb{F}^{[\![n]\!] \times [\![n]\!]})^{[\![r]\!]}$. Then $f_\ell(\vec{A}, \vec{x}) - f_\ell(\vec{B}, \vec{x})$ can be computed by a width $2n^2$, depth ℓ, degree $< r$ ROABP.*

Alternatively, when $\vec{x}$ is constant, and the matrices $\vec{M}$ are variable, then $f_\ell(\vec{M}, \vec{x})$ can be computed by a small ABP.

Lemma 3.3. *Assume the setup of Construction 3.1. Let $\vec{\alpha} \in \mathbb{F}^\ell$. Then $f_\ell(\vec{M}, \vec{\alpha})$ can be computed by a width n^2, depth ℓ ABP, and this ABP is constructable in $\mathsf{poly}(n, r, \ell)$ steps.*

Our next two lemmas highlight the connection between the polynomials in Construction 3.1 and the generators of the ring of invariants provided by Theorem 1.3. Namely, they show that the generators in the set $\mathcal{T}$ of Theorem 1.3 are faithfully encoded as coefficients of the polynomial $f_\ell(\vec{M}, \vec{x})$, when viewing this polynomial as lying in the ring $\mathbb{F}[\vec{M}][\vec{x}]$. Note here that we use the $\mathfrak{C}_{\vec{x}^{\vec{i}}}$ notation as defined in Subsection 1.4.

Given these two lemmas the proof of the reduction is similar to the one done in Mulmuley's paper. Nevertheless, for completeness we give the entire reduction.

Lemma 3.4. *Assume the setup of Construction 3.1. Then for $\vec{i} \in \mathbb{N}^\ell$, taking coefficients in $\mathbb{F}[\vec{M}][\vec{x}]$, $\mathfrak{C}_{\vec{x}^{\vec{i}}}(f_\ell(\vec{M}, \vec{x})) = \mathrm{trace}(M_{i_1} \cdots M_{i_\ell})$ if $\vec{i} \in [\![r]\!]^\ell$ and 0 otherwise.*

As the above lemma shows that $f_\ell(\vec{M}, \vec{x})$ encodes all of the generators $\mathcal{T}$, it follows that $\vec{A}$ and $\vec{B}$ agree on the generators $\mathcal{T}$ iff they agree on $f_\ell(\vec{M}, \vec{x})$.

Lemma 3.5. *Assume the setup of Construction 3.1. Let $\vec{A}, \vec{B} \in (\mathbb{F}^{[\![n]\!] \times [\![n]\!]})^{[\![r]\!]}$ and $\ell \geq 1$. Then $\mathrm{trace}(A_{i_1} \cdots A_{i_\ell}) = \mathrm{trace}(B_{i_1} \cdots B_{i_\ell})$ for all $\vec{i} \in [\![r]\!]^\ell$ iff $f_\ell(\vec{A}, \vec{x}) = f_\ell(\vec{B}, \vec{x})$, where this second equality is as polynomials in the ring $\mathbb{F}[\vec{x}]$.*

[3] There are some slightly better versions of this construction, as well as a way to more efficiently use the hitting sets of Theorem 1.11. However, these modifications make the presentation slightly less modular, and do not improve the results by more than a polynomial factor, so we do not pursue these details.

Hence, the polynomials $f_\ell(\vec{M}, \vec{x})$ capture the generators $\mathcal{T}$ of $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ thus in a sense capturing the entire ring $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ also, by Theorem 1.3.

Corollary 3.6. *Assume the setup of Construction 3.1. Let $\mathbb{F}$ be a field of characteristic zero. Let $\vec{A}, \vec{B} \in (\mathbb{F}^{[\![n]\!]\times[\![n]\!]})^{[\![r]\!]}$. Then $f(\vec{A}) = f(\vec{B})$ for all $f \in \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ iff $f_\ell(\vec{A}, \vec{x}) = f_\ell(\vec{B}, \vec{x})$ for all $\ell \in [n^2]$, where the second equality is as polynomials in the ring $\mathbb{F}[\vec{x}]$.*

Thus having reduced the question of whether $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ separates $\vec{A}$ and $\vec{B}$, to the question of whether some $f_\ell(\vec{M}, \vec{x})$ separates $\vec{A}$ and $\vec{B}$, we now seek to remove the need for the indeterminates $\vec{x}$. Specifically, we will replace them by the evaluation points of a hitting set, as shown in the next construction.

Construction 3.7. *Assume the setup of Construction 3.1. Let $\mathcal{H} \subseteq \mathbb{F}^{n^2}$ be a $t(n,r)$-explicit hitting set for width $\leq 2n^2$, depth n^2, degree $< r$ ROABPs. Define $\mathcal{T}_\mathcal{H} := \{f_\ell(\vec{M}, \vec{\alpha}) | \vec{\alpha} \in \mathcal{H}, \ell \in [n^2]\}$, where if $\ell < n^2$ we use the first ℓ variables of α for the values of $\vec{x}$ in the substitution.*

We now prove the main theorems, showing how to construct small sets of explicit separating invariants. We first do this for an arbitrary hitting set, then plug in the hitting set given in our previous work as stated in Theorem 1.11.

Theorem 3.8. *Assume the setup of Construction 3.7. Let $\mathbb{F}$ be a field of characteristic zero. Then $\mathcal{T}_\mathcal{H}$ is a set of size $n^2|\mathcal{H}|$ of homogeneous separating invariants with $\mathsf{poly}(t(n,r), n, r)$-explicit ABPs. That is, $\mathcal{T}_\mathcal{H} \subseteq \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$, and for any $\vec{A}, \vec{B} \in (\mathbb{F}^{[\![n]\!]\times[\![n]\!]})^{[\![r]\!]}$, $f(\vec{A}) \neq f(\vec{B})$ for some $f \in \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ iff $f'(\vec{A}) \neq f'(\vec{B})$ for some $f' \in \mathcal{T}_\mathcal{H}$, and each such f' is computed by an explicit ABP.*

As done in Mulmuley's Theorem 3.6, we can conclude that the ring of invariants is integral over the subring generated by the separating invariants. This uses the following theorem of Derksen and Kemper [DK02] (using the ideas of geometric invariant theory [MFK94]), which we only state in our specific case, but does hold more generally.

Theorem 3.9 (Theorem 2.3.12, Derksen and Kemper [DK02], stated by Mulmuley in Theorem 2.11). *Let $\mathbb{F}$ be an algebraically closed field of characteristic zero. Let $\mathcal{T}' \subseteq \mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ be a finite set of homogeneous separating invariants. Then $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ is integral over the subring S generated by $\mathcal{T}'$.*

Combining Theorem 3.8 and Theorem 3.9 yields the following corollary.

Corollary 3.10. *Assume the setup of Construction 3.7. Let $\mathbb{F}$ be an algebraically closed field of characteristic zero. Then $\mathbb{F}[\vec{M}]^{\mathrm{GL}_n(\mathbb{F})}$ is integral over the subring generated by $\mathcal{T}_\mathcal{H}$, a set of $n^2|\mathcal{H}|$ invariants with $\mathsf{poly}(t(n,r), n, r)$-explicit ABPs.*

Continuing with the dictionary of geometric invariant theory [MFK94], we can obtain the following deterministic black-box algorithm for testing of two orbit closures intersect.

Corollary 3.11. *Assume the setup of Construction 3.7. Let $\mathbb{F}$ be a field of characteristic zero. There is an algorithm, running in deterministic $\mathsf{poly}(n, r, t(n, r), |\mathcal{H}|)$-time, in the unit cost arithmetic model, such that given $\vec{A}, \vec{B} \in (\mathbb{F}^{[n]\times[n]})^{[r]}$, one can decide whether the orbit closures of $\vec{A}$ and $\vec{B}$ under simultaneous conjugation have an empty intersection. Further, this algorithm is "black-box", as it simply compares $f(\vec{A})$ and $f(\vec{B})$ for various polynomials f.*

Thus, the above results, Theorem 3.8, Corollary 3.10, and Corollary 3.11 give positive results to Questions 1.8, 1.6, and 1.9 respectively, assuming small explicit hitting sets for ROABPs. Plugging in the hitting sets results of Forbes and Shpilka [FS12] as cited in Theorem 1.11, we obtain Theorem 1.12 and Corollary 1.13.

However, using the hitting set of Theorem 1.11 does not allow us to deduce the efficient algorithm for orbit closure intersection claimed in Theorem 1.14 as the hitting set is too large. To get that result, we observe that deciding the orbit closure intersection problem does not require black-box PIT, and that white-box PIT suffices. Thus, invoking the white-box results of Raz and Shpilka [RS05], and the follow-up work by Arvind, Joglekar and Srinivasan [AJS09], we can get the desired result, as proven in the full version [FS13] of the paper.

Acknowledgments. The first author would like to thank Peter Bürgisser for explaining the work of Mulmuley, and the Complexity Theory week at Mathematisches Forschungsinstitut Oberwolfach for facilitating that conversation. He would also like to thank Sergey Yekhanin for the conversation that led to the results on non-closed orbit membership, and Scott Aaronson for some helpful comments.

The authors are grateful to Josh Grochow [Gro13] for bringing the works of Chistov-Ivanyos-Karpinski [CIK97], Sergeichuk [Ser00] and Belitskiĭ [Bel83] to their attention, and for explaining these works to them.

In an earlier version of the paper we mentioned Theorem 2.1 as a new observation. However, Pascal Koiran informed us that this was first observed by Malod and Portier [MP08].[4] We thank Pascal for turning our attention to this fact. We would also like to thank anonymous reviewers for their comments.

References

[AJS09] Arvind, V., Joglekar, P.S., Srinivasan, S.: Arithmetic Circuits and the Hadamard Product of Polynomials. In: FSTTCS, pp. 25–36 (2009) 535, 540

[ASS12] Agrawal, M., Saha, C., Saxena, N.: Quasi-polynomial hitting-set for set-depth-Δ formulas.. Electronic Colloquium on Computational Complexity (ECCC) 19(113) (2012)

[ASS13] Agrawal, M., Saha, C., Saxena, N.: Quasi-polynomial hitting-set for set-depth-Δ formulas. In: STOC, pp. 321–330. Full version in [ASS12] (2013) 527, 536

[4] The result is mentioned immediately after Theorem 4 in [MP08].

[Bel83] Belitskiĭ, G.R.: Normal forms in a space of matrices. In: Analysis in Infinite-dimensional Spaces and Operator Theory, pp. 3–15 (1983) (in Russian) 540

[Ber84] Berkowitz, S.J.: On computing the determinant in small parallel time using a small number of processors. Inf. Process. Lett. 18(3), 147–150 (1984) 534

[CIK97] Chistov, A.L., Ivanyos, G., Karpinski, M.: Polynomial time algorithms for modules over finite dimensional algebras. In: Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC), pp. 68–74 (1997) 540

[CLO07] Cox, D., Little, J., O'Shea, D.: Ideals, varieties, and algorithms, 3rd edn. Undergraduate Texts in Mathematics. Springer, New York (2007) 528

[DK02] Derksen, H., Kemper, G.: Computational invariant theory. In: Invariant Theory and Algebraic Transformation Groups, I. Springer, Berlin (2002); Encyclopaedia of Mathematical Sciences, vol. 130 528, 532, 539

[DL78] DeMillo, R.A., Lipton, R.J.: A probabilistic remark on algebraic program testing. Inf. Process. Lett. 7(4), 193–195 (1978) 530

[For86] Formanek, E.: Generating the ring of matrix invariants. In: Ring theory (Antwerp, 1985). Lecture Notes in Math., vol. 1197, pp. 73–82. Springer, Berlin (1986) 531

[FS12] Forbes, M.A., Shpilka, A.: Quasipolynomial-time identity testing of noncommutative and read-once oblivious algebraic branching programs. Electronic Colloquium on Computational Complexity (ECCC) 19, 115 (2012) 527, 534, 535, 536, 537, 540

[FS13] Forbes, M.A., Shpilka, A.: Explicit noether normalization for simultaneous conjugation via polynomial identity testing. Electronic Colloquium on Computational Complexity (ECCC) 20, 33 (2013) 536, 537, 540

[GKKS13] Gupta, A., Kamath, P., Kayal, N., Saptharishi, R.: Arithmetic circuits: A chasm at depth three. Electronic Colloquium on Computational Complexity (ECCC) 20(26) (2013) 536

[Gro13] Grochow, J.: Private communication (2013) 540

[IKW02] Impagliazzo, R., Kabanets, V., Wigderson, A.: In search of an easy witness: exponential time vs. probabilistic polynomial time. J. Comput. Syst. Sci. 65(4), 672–694 (2002) 528

[IW97] Impagliazzo, R., Wigderson, A.: P=BPP if E requires exponential circuits: Derandomizing the XOR lemma. In: STOC, pp. 220–229 (1997) 528

[KI04] Kabanets, V., Impagliazzo, R.: Derandomizing polynomial identity tests means proving circuit lower bounds. Computational Complexity 13(1-2), 1–46 (2004) 528

[KP00] Kraft, H., Procesi, C.: Classical invariant theory, a primer. Lecture Notes (2000) 531

[MFK94] Mumford, D., Fogarty, J., Kirwan, F.: Geometric invariant theory. Ergebnisse der Mathematik und ihrer Grenzgebiete (2) [Results in Mathematics and Related Areas (2)], 3rd edn., vol. 34. Springer, Berlin (1994) 532, 533, 539

[MM82] Mayr, E.W., Meyer, A.R.: The complexity of the word problems for commutative semigroups and polynomial ideals. Adv. in Math. 46(3), 305–329 (1982) 528

[MP08] Malod, G., Portier, N.: Characterizing valiant's algebraic complexity classes. J. Complexity 24(1), 16–38 (2008) 537, 540

[Mul12a] Mulmuley, K.: Geometric complexity theory V: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of Noether's normalization lemma. In: FOCS, pp. 629–638 (2012) 527, 528

[Mul12b] Mulmuley, K.: Geometric complexity theory V: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of Noether's normalization lemma. arXiv, abs/1209.5993 (2012); Full version of the FOCS 2012 paper 528

[Mul12c] Mulmuley, K.: Geometric complexity theory V: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of Noether's normalization lemma (2012), http://techtalks.tv/talks/geometric-complexity-theory-v-equivalence-between-blackbox-derandomization-of-polynomial-identity-testing-and-derandomization-of/57829/ 529

[Nis91] Nisan, N.: Lower bounds for non-commutative computation. In: STOC, pp. 410–418 (1991) 534

[Pro76] Procesi, C.: The invariant theory of $n \times n$ matrices. Advances in Math. 19(3), 306–381 (1976) 531

[Raz74] Razmyslov, J.P.: Identities with trace in full matrix algebras over a field of characteristic zero. Izv. Akad. Nauk SSSR Ser. Mat. 38, 723–756 (1974) 531

[RS05] Raz, R., Shpilka, A.: Deterministic polynomial identity testing in non-commutative models. Computational Complexity 14(1), 1–19 (2005) 535, 536, 540

[Sax08] Saxena, N.: Diagonal circuit identity testing and lower bounds. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 60–71. Springer, Heidelberg (2008) 527, 536

[Sch80] Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. Assoc. Comput. Mach. 27(4), 701–717 (1980) 530

[Ser00] Sergeichuk, V.V.: Canonical matrices for linear matrix problems. Linear Algebra Appl. 317(1-3), 53–102 (2000) 540

[SSS11] Saha, C., Saptharishi, R., Saxena, N.: A Case of Depth-3 Identity Testing, Sparse Factorization and Duality. Computational Complexity 22(1), 39–69 (2013) 536

[SV09] Shpilka, A., Volkovich, I.: Improved polynomial identity testing for read-once formulas. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX and RANDOM 2009. LNCS, vol. 5687, pp. 700–713. Springer, Heidelberg (2009) 527, 536

[SY10] Shpilka, A., Yehudayoff, A.: Arithmetic circuits: A survey of recent results and open questions. Foundations and Trends in Theoretical Computer Science 5(3-4), 207–388 (2010) 529, 530, 534

[Val79] Valiant, L.G.: Completeness classes in algebra. In: STOC, pp. 249–261 (1979) 534

[Zip79] Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Ng, E.W. (ed.) EUROSAM 1979. LNCS, vol. 72, pp. 216–226. Springer, Heidelberg (1979) 530

Testing Membership in Counter Automaton Languages*

Yonatan Goldhirsh and Michael Viderman

Department of Computer Science
Technion — Israel Institute of Technology
Haifa, 32000, Israel
{jongold,viderman}@cs.technion.ac.il

Abstract. Alon et al. (SICOMP 2001) showed that all regular languages are testable with a constant number of queries. On the other hand, they also showed that some context free languages require $\Omega(\sqrt{n})$ queries to test. Following this, Alon et al. suggested the problem of classifying the context free languages that are testable with a constant number of queries.

We make progress towards the solution of this problem. Our main result is that languages accepted by *weak counter automata* are testable with a constant number of queries. A counter automaton is a pushdown automaton with a single stack symbol, effectively a non-negative counter that the automaton may compare to zero. It is *weak* if the set of possible transitions with a zero counter is a subset of the possible transitions with a positive counter. Note that this restriction is essential, since Lachish et. al. (CC 2008) proved that there exist counter automaton languages requiring $\Omega(\mathrm{polylog}(n))$ queries to test.

1 Introduction

Property testing is a relaxation of the standard decision problem. Instead of requiring the algorithm to distinguish inputs possessing some property from those who don't, a property testing algorithm is only required to distinguish inputs that have a certain property from those that are *far* from having the property. More formally, a property testing algorithm for a property P is given query access to some input object x and a distance parameter $\epsilon > 0$. The algorithm must accept with high probability if x has the property P, and reject with high probability if x is ϵ-far from any object that has the property P. In this context ϵ-far refers to some fixed distance function, usually implying that more than an ϵ-fraction of x must be changed so that it will have the property P.

The main complexity measure in property testing is the *query complexity* of the algorithm, namely, the number of queries it performs. Ideally, we want the query complexity to depend only on ϵ, and not on the size of the input. Properties admitting algorithms with query complexity that depends only on ϵ are called *testable* (sometimes emphasizing "with a constant number of queries"). An important distinction is made between *1-sided testers* which must accepts inputs in P, and *2-sided testers*, which don't have this further restriction.

* The research leading to these results has received funding from the European Union's - Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 202405 (PROPERTY TESTING).

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 543–558, 2013.

The notion of property testing was first defined by Rubinfeld and Sudan [16,17] in an algebraic setting, and later in a combinatorial setting by Goldreich, Goldwasser and Ron [7]. Since then much research was done both on finding the optimal query complexity for specific properties, and on the major problem of characterizing those properties that have a property tester of constant query complexity (or at least finding large families of such properties). For more information, see the surveys [3,15,14].

Property testing of low complexity languages An important challenge in this respect is to study the relationship between property testing and other notions of complexity. Are "low complexity" objects also testable? The first work to tackle this problem was by Alon et. al. [1], showing that all regular languages are 1-sided testable with a constant number of queries. Alon et. al. further showed that such a strong result does not hold for context free languages in general. In particular, Alon et. al. considered Dyck languages, the languages of properly balanced parentheses of one or more types[1]. They showed that the Dyck language for one type is testable, yet the Dyck language for two types requires $\Omega(\log n)$ queries[2]. Alon et. al. also showed that the concatenated palindrome language, defined by the set $\{uu^rvv^r : u, v \in \Sigma^*\}$, requires $\Omega(\sqrt{n}/\epsilon)$ queries to test, even for 2-sided testers. Parnas et. al. [13] showed that while these languages are not testable with a constant number of queries, they do admit property testers of sublinear query complexity, proving an upper bound of $\tilde{O}(n^{2/3}/\epsilon^3)$ for testing all Dyck languages, and an upper bound of $\tilde{O}(\sqrt{n}/\epsilon)$ for testing the concatenated palindrome language.

In their paper, Alon et. al. pose the problem of finding sufficient conditions for a context free languages to be testable [1, Section 5]. A negative result in this regard was proved by Lachish et. al. [11], showing that even if we restrict our attention to counter automata, which are pushdown automata with only one stack symbol, we still encounter untestable languages. Specifically, they give a language accepted by a counter automaton yet requiring $\Omega(\text{polylog}(n))$ to test, even for 2-sided testers. This leads us to further restrict the computational model. In the language given by Lachish et. al. some of the transitions are available only when the counter equals zero. This leads to the lower bound because of the difficulty that any testing algorithm will have with determining when the counter should equal zero (had we been following an accepting computation for the input word). We alleviate this problem by including another restriction — the set of possible transitions with a zero counter must be a subset of the possible transitions with a positive counter (see Definition 2). We call such an automaton a *weak counter automaton*, and the focal point of this work is proving that all languages accepted by such automata are testable with a constant number of queries. Obviously, regular languages are accepted by such automata, furthermore, in the full version of the paper we show that there is a rich family of non-regular languages accepted by weak counter automata.

It is also important to note that there has been a considerable body of work attempting to prove testability (or hardness of it) for *nonuniform* models of computation, such as bounded width branching programs [12,6] and formula [8,4,2,5].

[1] The Dyck language for m types of parentheses is defined by setting m pairs of parentheses symbols, and defining the languages to be the set of all property balanced strings over these parentheses symbols.

[2] Note that while untestable, the Dyck language for any number of types can be accepted by a pushdown automaton with a **single** state.

Organization of the paper. In the following section we provide background regarding property testing and counter automata. In Section 3 we state our main result. Then, in Section 4 we give an overview of the central concepts underlying our testing algorithm and explain basic ideas behind this work. In particular, in Section 4.3 we give a simplified overview of the testing algorithm, present some technical obstacles we meet in this paper and describe the suggested solutions. We present our testing algorithm in Section 6.

The proofs of the main result and the corresponding lemmas and claims are deferred to the full version of the paper.

2 Preliminaries

We set $\mathbb{N}$ to be the non-negative integers and $\mathbb{N}^+$ the the strictly positive integers. Let $[n]$ be the set $\{1, \ldots, n\}$. For $n_1, n_2 \in \mathbb{N}$ we write $n_1 | n_2$ if n_2 divisible by n_1. For $w \in \{0,1\}^n$ we let $\mathrm{supp}(w) = \{i \in [n] \mid w_i \neq 0\}$ and $|w| = |\mathrm{supp}(w)|$. For any set A, let $\mathcal{P}(A) = \{A' \mid A' \subseteq A\}$ be its power set. We define the *distance* between two words $x, y \in \{0,1\}^n$ to be $\Delta(x,y) = |\{i \mid x_i \neq y_i\}|$ and the relative distance to be $\delta(x,y) = \frac{\Delta(x,y)}{n}$.

For $w \in \{0,1\}^n$ and $\mathcal{L} \subseteq \{0,1\}^*$, let $\delta(w, \mathcal{L}) = \min_{p \in \mathcal{L} \cap \{0,1\}^n} \{\delta(w,p)\}$ denote the relative distance of w from $\mathcal{L}$. If $\delta(x, \mathcal{L}) \geq \epsilon$, we say that w is ϵ-far from $\mathcal{L}$ and otherwise w is ϵ-close to $\mathcal{L}$.

2.1 Property Testers

In this paper, oracle access to a word $w \in \{0,1\}^n$ is a function mapping $i \mapsto w_i$ for every $i \in [n]$.

Definition 1. [ϵ-tester] *Let $\mathcal{L} \subseteq \{0,1\}^*$. An ϵ-tester T is a randomized algorithm that given $\epsilon > 0$, an integer n and oracle access to a word $w \in \{0,1\}^n$ performs $q = q(\epsilon, n)$ queries to w. If $w \in \mathcal{L}$, then T accepts with probability at least $2/3$. If $\delta(w, \mathcal{L}) > \epsilon$, then T rejects with probability at least $2/3$.*

The parameter ϵ is called the distance *parameter and the function q is the* query complexity *of the ϵ-tester. $\mathcal{L}$ is said to be* testable *if q is independent of the word length n. We say that T has* one-sided error *if it always accepts inputs $w \in \mathcal{L}$, and otherwise it has* two-sided error. *If the choice of the queries used by T is independent of the their values we say T is* non-adaptive *and otherwise we say it is* adaptive.

If such an algorithm exists for a propety P with query complexity that only depends on ϵ, we say that P is *testable*.

2.2 Weak Counter Automata

We only introduce facts regarding automata as needed for our purposes. For the basic facts regarding automata and regular languages, please refer to any standard textbook on formal language theory, such as [9]. For more information regarding counter automata

see [18]. Note that in the following definition, we allow a nondeterministic transition function.

Let $IsZero : \mathbb{N} \longrightarrow \{ZERO, NZERO\}$ be such that $IsZero(i) = ZERO$ if $i = 0$ and $NZERO$ otherwise.

Definition 2. [Counter Automaton, Weak Counter Automaton] *A* counter automaton M *over* $\{0,1\}$ *is defined by a tuple* $M = (V, q_1, F, \delta)$*, where* $V = \{q_1, \ldots, q_m\}$ *is a set of states,* $q_1 \in V$ *is the initial state,* $F \subseteq V$ *is a set of accepting states and* $\delta : V \times \{0,1\} \times \{ZERO, NZERO\} \longrightarrow \mathcal{P}(V \times \{-1, 0, +1\})$ *is the transition function. The transition function maps a state, current tape symbol and counter status to a set of pairs of the form (state, change in counter value). If additionally for every* $q \in V$ *and* $\sigma \in \{0,1\}$ *we have that* $\delta(q, \sigma, ZERO) \subseteq \delta(q, \sigma, NZERO)$*, we say that the counter automaton is* weak.

For a weak counter automaton M and word $w \in \{0,1\}^*$ define the relation $\rightarrow_w^M$ on pairs from $V \times \mathbb{N}$ by induction on the word structure. Let γ denote the empty word. First, for any $\mathrm{num} \in \mathbb{N}$ and $q \in V$, $(q, \mathrm{num}) \rightarrow_\gamma^{\mathrm{M}} (\mathrm{q}, \mathrm{num})$.

Next, assume that $(q, \mathrm{num}) \rightarrow_{\mathrm{u}}^{\mathrm{M}} (\mathrm{p}, \mathrm{num}')$ for some $u \in \{0,1\}^*$, $p, q \in V$ and $\mathrm{num}, \mathrm{num}' \in \mathbb{N}$. If $b \in \{0,1\}, p' \in V$ and $c \in \{-1, 0, +1\}$ are such that $(p', c) \in \delta(p, b, IsZero(\mathrm{num}'))$ and $\mathrm{num}' + \mathrm{c} \geq 0$ then $(q, \mathrm{num}) \rightarrow_{\mathrm{ub}}^{\mathrm{M}} (\mathrm{p}', \mathrm{num}' + \mathrm{c})$.

Let $w \in \{0,1\}^*$ be a word and recall that $q_1 \in V$ is the initial state. We say that w is *accepted* by M if there exist $p \in F$ and $\mathrm{num} \in \mathbb{N}$ such that $(q_1, 0) \rightarrow_w^M (p, \mathrm{num})$. Given the automaton M we denote by $L(M)$ the language accepted by the automaton M, i.e.,

$$L(M) = \left\{ w \in \{0,1\}^* | M \text{ accepts the word } w \right\}.$$

When the automaton M is clear from context, we may omit it and write $(q, l_1) \rightarrow_w (p, l_2)$ and say that (p, l_2) is *reachable* from (q, l_1). When we use a pair of the form $(q, \cdot)$, it is to be understood as "there exists $l \in \mathbb{N}$ such that when substituting l for $\cdot$ the relation holds". For example, $(q, \cdot) \rightarrow_w (p, \cdot)$ if there exist $l_1, l_2 \in \mathbb{N}$ such that $(q, l_1) \rightarrow_w (p, l_2)$.

3 Main Result

In the following theorem we state the main result of this paper.

Theorem 1 (Main Theorem). *Let* $M = (V, q_1, F, \delta)$ *be a weak counter automaton. Then for every* $n \in \mathbb{N}$*, the property* $L(M) \cap \{0,1\}^n$ *is testable by a two-sided* ϵ*-tester.*

The full proof of Theorem 1 is omitted from this extended abstract.

Theorem 1 expands on the main result of [1] since every regular language can be accepted by a weak-counter automaton. On the other hand, weak counter automata can also accept interesting non-regular languages, such as the language of prefixes of balanced parenthesis strings.

Remark 1. Theorem 1 is not a direct generalization of the regular languages tester of [1], since the weak counter automaton languages tester is 2-sided, while the regular languages tester is 1-sided.

This is the best that can be expected, since some languages acceptable by weak counter automata do not admit a 1-sided tester:

Denote by $pref(x)$ the set of prefixes for a string x, and denote by $\sharp_0(x)$ and $\sharp_1(x)$ the number of 0s and 1s in x respectively.

Now consider the language $L = \{x \in \{0,1\}^n | \forall w \in pref(x), \sharp_0(w) \geq sharp_1(w)\}$, which can be accepted by a weak counter automaton. That is, the language of strings where all prefixes have more 0s than 1s. The string $0^{n/4}1^{3n/4}$ is $\frac{1}{4}$-far from L, but any witness of size smaller than $n/4$ can be completed to a string in L. Therefore any 1-sided tester for L must use at least $n/4$ queries.

4 Overview

In this section we introduce the central concepts underlying the algorithm and its proof, and survey the general method it employs.

4.1 Automata and Graphs

Every counter automaton defines a nondeterministic finite automaton by "ignoring" the counter. Formally, if $M = (V, q_1, F, \delta)$ is a counter automaton then it corresponds to the nondeterministic finite automaton $M^R = (V, q_1, F, \delta^R)$ where if $(q, c_2) \in \delta(p, a, c_1)$ then $q \in \delta^R(p, a)$.

Next, every finite automaton $M = (V, q_1, F, \delta)$ defines a directed graph $G(M)$ over the set of vertices $V(G(M)) = V$ with the set of edges being $E(G(M)) = \{(q_i, q_j) \mid \exists b \in \{0,1\} : \ q_j \in \delta(q_i, b)\}$. Note that we allow self loops in this graph.

Every word $w \in \{0,1\}^n$ defines several paths in the graph, each corresponding to one of the possible computation paths for w in the automaton.

For a given automaton $M = (V, q_1, F, \delta)$ with a graph $G = G(M)$, we denote by $C(G)$ the graph of strongly connected components of G. Recall that a strongly connected component of a directed graph $G = (V, E)$ is a maximal (by inclusion) subset $U \subseteq V$ such that for every $u, v \in V$ there is a path from u to v and from v to u. The graph of strongly connected components of G is defined with the vertex set being all strongly connected components of G, and an edge leads from a component C_i to a component C_j whenever there exist $u \in C_i, v \in C_j$ such that (u, v) is an edge in G. Note that some of the vertices of $C(G)$ may represent a single vertex of G with no self loops. All other components have non empty paths inside them and will be called *truly connected*. From now on we reserve k for the number of vertices of $C(G)$, note that we can always bound $k \leq |V|$.

We may assume that all vertices of G are reachable from the initial state q_1. Therefore in $C(G)$ there is a directed path from the component C_1, which contains q_1, to every other component. Recall that $C(G)$ is always an acyclic graph.

4.2 Runs in Words

In this section, let $w \in \{0,1\}^n$ and $M = (V, q_1, F, \delta)$ be the counter automaton. For $i \leq j \in [n]$, we denote by $w[i, j]$ the consecutive substring of w starting at index i and ending at index j.

Note that the definitions in the section also depend on n, the length of the input word. Since n is fixed throughout the discussion, as a property is a subset of $\{0,1\}^n$, we keep this dependence implicit.

Definition 3. [Feasible and infeasible runs] *A sub-word (run) $w[i,j]$ for $i \leq j \in [n]$ is called* feasible *for M, if there exist states $q, q' \in V$, an accepting state $p \in F$, $x_1 \in \{0,1\}^{i-1}$ and $x_2 \in \{0,1\}^{n-j}$ such that the following all hold:*

- $(q_1, \cdot) \rightarrow_{x_1} (q, \cdot)$
- $(q, \cdot) \rightarrow_{w[i,j]} (q', \cdot)$
- $(q', \cdot) \rightarrow_{x_2} (p, \cdot)$

Otherwise, $w[i,j]$ is called infeasible.

One of the possible witnesses that a word $w \notin L(M)$ is an infeasible run. Unfortunately, the absence of such witnesses is not enough to imply a word $w \in L(M)$, since it does not take into account the counter values. This leads to the next definition.

Definition 4. [Value of a run] *Let $w \in \{0,1\}^n$ and $i \leq j \in [n]$ such that $w[i,j]$ is a feasible run. We say that* $\mathrm{val_M(w[i,j])} = \mathrm{v}$ *if* $\mathrm{v} \in \mathbb{Z}$ *is the maximal value such that there exists $w' \in L(M) \cap \{0,1\}^n$, $c \in \mathbb{N}$, $q, q' \in V$ and $p \in F$ satisfying:*

- $w'[i,j] = w[i,j]$,
- $(q_1, \cdot) \rightarrow_{w'[1,i-1]} (q, c)$,
- $(q, c) \rightarrow_{w'[i,j]} (q', c + \mathrm{v})$,
- $(q', c + \mathrm{v}) \rightarrow_{w'[j+1,n]} (p, \cdot)$.

Note that in particular, we defined $\mathrm{val_M(w)}$. *If $w[i,j]$ is infeasible, we set* $\mathrm{val_M(w[i,j])} = 0$[3].

Note that $-(j-i+1) \leq \mathrm{val_M(w[i,j])} \leq (\mathrm{j} - \mathrm{i} + 1)$.

Informally, the value of a feasible run $w[i,j]$ is the most positive change to the counter that could happen while the automaton is reading $w[i,j]$ in a computation that can conclude in an accepting state, assuming a high enough initial counter value.

Another important concept is that of the word prefix which necessitates the most negative counter change in a computation path on a word.

Definition 5. [Minimal prefix] *We define* $\mathrm{minPref_M(w)} = \min_{\mathrm{j} \in [\mathrm{n}]} \{\mathrm{val_M(w[1,j])}\}$. *For a sequence of runs* $w[0,h-1], w[h,2h-1], \ldots w[n-h,n]$, *we define the related quantity* $\mathrm{minPref_M(w,h)} = \min_{\mathrm{i} \in [\mathrm{t}]} \left\{ \sum_{\mathrm{j} \leq \mathrm{i}} \mathrm{val_M(w[(j-1)h, jh-1])} \right\}$.

4.3 Algorithm Overview

In this subsection we provide a coarse and simplified overview of the testing algorithm. Our main testing algorithm (Algorithm 1) receives as input a weak counter automaton

[3] While it does not seem natural to assign a value to an infeasible run, it's technically convenient in the proof.

M, a word w and a distance parameter $\epsilon > 0$. Algorithm 1 queries a constant number bits from w. If $w \in L(M)$ it accepts with high probability, if w is ϵ-far from $L(M)$ it rejects with high probability. This algorithm invokes an auxiliary $\mathrm{Approximator}$ as a subroutine.

Assume $M = (V, q_1, F, \delta)$ is a weak counter automaton and $w \in \{0,1\}^n$ is an input word. We shall test whether $w \in L(M)$. If indeed $w \in L(M)$, there is some computation path in M certifying that, starting at q_1 and ending in some state in F. This path can be decomposed according to the strongly connected components of $G(M)$, where we will also specify the first and last state visited in every component, and the indices in $[n]$ during which the computation visited each component. Note that since $C(G(M))$ is acyclic, every component can be visited at most once.

This decomposition allows us to reduce the testing of $w \in L(M)$ to that of testing membership of subwords of w in "sub-automata" of M, where these "sub-automata" define strongly connected graphs (see Section 5.2). To execute this task, Algorithm 1 invokes $\mathrm{Approximator}$ on the corresponding "sub-automata" of M and subwords of w. That is, the input to $\mathrm{Approximator}$ is a corresponding subword, a distance parameter and a sub-automaton. Given automata that define strongly connected graphs, $\mathrm{Approximator}$ samples a few short consecutive subwords (runs) in w in order to find infeasible runs and approximate the value (as in Definition 4) and minimal prefix (Definition 5) of the input subword.

The output of $\mathrm{Approximator}$ is true/false (i.e., whether a given subword corresponds to a possible computation path in the given sub-automaton), estimated minimal prefix and the value of the input subword. Assuming that $\mathrm{Approximator}$ outputs true and that these approximations are good enough, they can be added together to determine whether this is truly an accepting computation path for $w \in L(M)$. Note that these approximations do not necessarily correspond to the original computation path considered, but rather to a path which follows a similar decomposition.

Unfortunately, we cannot expect to be supplied with such a decomposition. We would like to iterate over all possible decompositions, but their number depends on n. Since we are looking for an algorithm with query complexity independent of n, we prove that there is a set of decompositions which "approximate" all other decompositions, and that this set is of size independent of n. Thus our main testing algorithm (Algorithm 1) iterates over this small set of decompositions. If at least one such decomposition passes all tests then Algorithm 1 accepts w, otherwise it rejects.

Now we proceed to explain some technical obstacles and suggested solutions.

High precision of the counter approximations. We need to prove that whenever $w \in \{0,1\}^n$ is ϵ-far from $L(M)$, Algorithm 1 rejects with probability at least $2/3$. Equivalently, if Algorithm 1 rejects with probability less than $2/3$, then at most ϵn bits in w can be modified to obtain a word in $L(M)$. Since $\epsilon > 0$ can be arbitrary small, the precision of the counter approximations made by $\mathrm{Approximator}$ should be arbitrary high (depending on ϵ). Therefore, the number of runs we select depends on ϵ to make the approximations sufficiently precise. Moreover, since parts of different runs might require a modification (to "glue" a sequence of runs together), the length of the queried runs depends on ϵ as well.

Reduction to sub-automata and subwords. Assume we are given a decomposition to "sub-automata" of M and subwords of w. We need to verify whether this decomposition describes an accepting run of w in M. If M were a finite automaton, this task would be reduced to just picking a sufficiently large number of runs with respect to every sub-automata and checking whether at least one of them is infeasible (as in [1]). Indeed, in $\mathrm{Approximator}$ we take this approach. However, M is a weak counter automaton, which means that even if all runs of w are feasible, it might be that w is very far from being in $L(M)$. For example, consider the case when the entire word w is feasible for M, but almost all transitions in the proposed computation path for w in M decrease the counter. Notice that whenever a counter reaches the value 0 and all possible transitions will decrease the counter, the execution of M on w halts and w is rejected.

Therefore, for every corresponding subword of w we need to estimate the counter update caused by reading this subword by the corresponding sub-automaton of M. To explain this, assume that a counter has (a sufficiently large) value v when the computation enters the corresponding sub-automaton, and it has a value v' when the computation exits this sub-automaton. In this case, the update is $v' - v$. In particular, the update of the counter can be positive or negative. These updates can be summed up to verify whether there exists a witness that a current decomposition cannot describe an accepting computation path of w on M. For example, suppose we have 3 sub-automata in the decomposition. Initially the counter is 0. The first sub-automata implies an update of $+\frac{n}{10}$, the second sub-automata implies an update of $-\frac{n}{5}$ and the third sub-automata implies an update of $+\frac{n}{5}$. In this case, our prediction (that is made by $\mathrm{Approximator}$) says that after exiting the second sub-automata, the counter's value should be negative ($\frac{n}{10} - \frac{n}{5}$), which is impossible. Hence, the given decomposition cannot describe an accepting computation path of w on M.

Similar problems may arise inside a single sub-automaton. Suppose a first sub-automaton in the decomposition makes a positive update on the counter's value, however, some prefix of the suggested traversal path inside this sub-automaton has negative value. For example, first 10 steps of the suggested traversal path increase the counter's value by 10, but the following 20 steps decrease the counter's value by 20 (i.e., in the middle of this path counter's value becomes -10). In this case, even if the counter's update of the entire computation path is positive, this computation path is impossible since a counter's value cannot be negative. This is why we also need to approximate the minimal prefix (Definition 5) of a subword with respect to the corresponding sub-automata.

Complications near zero. As we mentioned, $\mathrm{Approximator}$ queries only a constant number of bits from a given subword. Therefore, it is plausible that the values estimated by this algorithm are only close to being correct, but not completely correct. In particular, if $\mathrm{Approximator}$ says that a counter's value is near 0 at some point, it can be a "good" case, where a suggested computation path results in a nonnegative value in the counter, but it can be also a "bad" case, where a suggested computation path results in a negative value in the counter. In the first case we should accept, but in the second case it seems that we must reject. The problem is that we cannot distinguish between two cases using constant query complexity.

Our solution is to accept situations, where $\mathrm{Approximator}$ shows that the counter's value is near 0. However, if a suggested computation implies a "slightly" negative

counter's value, our main testing algorithm (Algorithm 1) accepts with high probability and thus we are required to prove that w is close to $L(M)$. That is, we need to show how to modify a small number of bits in w to fit some accepting computation path in the sub-automaton.

This is where the weakness of the counter automaton comes into play. Informally, a counter automaton being weak means that if we were to suddenly increase the counter value mid-computation, an accepting computation path will still be able to accept. This is simply because a non-rejecting transition may not depend on the counter being zero. Therefore, what we'd like to do is simulate this increase in counter value with a small change to the input word. The tool we wish to employ here is reachable positive cycles in the automaton (see Definition 7). If we have such a cycle, we can hope to "ramp up" the counter using it, and thus do away with small errors in the $\mathrm{Approximator}$'s approximation. But what if we have no such cycles? As it turns out, if no sub-automata has a reachable positive cycle, then the counter value is bounded by a function of the automaton's size, and the computation (and testing) can be reduced to that of a finite automaton (see the full version of the paper).

Now we are left to contend with the subtlety of using the reachable positive cycles. Ideally, we'd like to replace some of the sub-word in the sub-automaton where this reachable cycle appears with a repeated iteration over the positive cycle. Unfortunately, it might be that this sub-word is not long enough for us to increase the counter to our desired value. A simple fix for that is changing the transitions between sub-automata slightly and "borrow" computation time from a later sub-automata for increasing the counter value. This simple fix can be problematic though, since we must make sure that we leave enough computation time in the later sub-automata we borrowed from so that we could find a word to get though it.

Handling all of these issues requires a very careful analysis, which appears in the full version of the paper, where we develop a systematic way of "ramping up" the counter in a way that results in an accepting computation.

We follow with a subsection formalizing the decomposition of a computation path. The algorithms and their proofs are fully developed in the Appendix.

4.4 Formalizing the Details

We now formally define this notion of decomposition.

Admissible Path and 4-Tuple

A computation path for a word naturally defines a sequence of triplets we denote $(C_1, q_1, l_1), (C_2, q_2, l_2), \ldots, (C_f, q_f, l_f)$, where for every $j \in [f]$ we have that C_j is a strongly connected component, q_j is the first state in C_j that was reached in this path and l_j is the value stored in the counter when this path reached q_j for the first time. Triplets will be categorized according to whether they have a positive cycle that can be used by the computation. We define this formally:

Definition 6. [Reachable states and cycles] *Let $(v_1, \ldots, v_d, v_1)$ be a cycle in $G = G(M)$. We say that the cycle is reachable including the walk from (q, l), if there exist*

$w \in \{0,1\}^*$, $l_1, l_2, \ldots, l_d, l_{d+1} \in \mathbb{N}$ *and* $a_1, a_2, \ldots, a_d \in \{0,1\}$ *such that*

$$(q,l) \to_w (v_1,l_1) \to_{a_1} (v_2,l_2) \to_{a_2} (v_3,l_3) \to_{a_3} \ldots (v_d,l_d) \to_{a_d} (v_1,l_{d+1})$$

Finally, we say that for $q, q' \in V(G)$ *and* $l, l' \in \mathbb{N}$ *it holds that* (q',l') *is* h*-reachable from* (q,l) *on* $w \in \{0,1\}^*$ *if* $0 \leq l, l' \leq h$ *and there exist* $a_1, a_2, \ldots, a_{d-1} \in \{0,1\}$, $0 \leq l_1, l_2, \ldots, l_d \leq h$ *and* $v_1, v_2, \ldots, v_d \in V$ *such that* $l_1 = l$, $l_d = l'$, $v_1 = q$, $v_d = q'$ *and*

$$(v_1,l_1) \to_{a_1} (v_2,l_2) \to_{a_2} (v_3,l_3) \ldots \to_{a_{d-1}} (v_d,l_d),$$

i.e., during this travel a counter value is always between 0 *and* h*. We say that* (q',l') *is* h*-reachable from* (q,l) *if there exists* $w \in \{0,1\}^*$ *such that* (q',l') *is* h*-reachable from* (q,l) *on* w.

Note that a cycle being reachable including the walk implies not only that the computation may arrive at the cycle, but that it also may traverse it fully. It might be the case that a cycle is reachable, but the counter is too low to complete a computation over the cycle.

Definition 7. [Cycles and degenerate components] *Let* $(v_1, \ldots, v_k, v_1)$ *be a directed cycle inside the strongly connected component* C *in the graph* $G(M)$*. If there exist* $a_1, \ldots, a_k \in \{0,1\}$ *and* $c_1, \ldots, c_k, c_{k+1} \in \mathbb{N}$ *such that* $(v_1,c_1) \to_{a_1} (v_2,c_2) \to_{a_2} \ldots (v_k,c_k) \to_{a_k} (v_1,c_{k+1})$ *and additionally* $c_{k+1} > c_1$ *we say that this cycle is* positive.

Let C *be some strongly connected component and let* $p \in C$*. The triplet* (C,p,l) *is* degenerate *if no positive cycle in* C *is reachable including the walk from* (p,l) *(i.e., either there are no positive cycles in* C *at all or they are not reachable including the walk from* (p,l)*). Otherwise, this triplet is* non-degenerate.

Recall that a computation path for w on M defines a sequence of triplets

$$(C_1,q_1,l_1),(C_2,q_2,l_2),\ldots,(C_f,q_f,l_f).$$

We will be interested in the first non-degenerate triplet in this sequence. We will later prove that the computation preceding the first non-degenerate triplet can be reduced to that of a regular language (see the full version of the paper), this will allow us to reduce the problem of testing that subword to that of testing membership in a regular language as well.

4-tuple. We consider 4-tuples (A,P,Π,l), where $A = (C_{i_1},\ldots,C_{i_t})$ is a path of strongly connected components in $C(G)$, $P = (p_j^1,p_j^2)_{j=1}^t$ is an ordered list of vertex pairs such that $p_j^1, p_j^2 \in C_{i_j}$, $\Pi = (n_j)_{j=1}^{t+1}$ is a list of natural numbers and $l \in \mathbb{N}$.

Admissible 4-tuple. Fix any 4-tuple (A,P,Π,l). We will be interested in 4-tuples that may arise from a computation path in M. We call a path A in $C(G)$ *admissible*, if it starts at some strongly connected component C_{i_1} and ends at a component with an accepting state. Given an admissible path $A = (C_{i_1},\ldots,C_{i_t})$ in $C(G)$, a sequence $P = (p_j^1,p_j^2)_{j=1}^t$ of pairs of vertices of G (states of M) is an *admissible* sequence of portals if it satisfies the following restrictions:

1. (p_1^1, l) is $(|V|-1)$-reachable from $(q_1, 0)$;
2. $p_j^1, p_j^2 \in C_{i_j}$ for every $1 \leq j \leq t$;
3. $p_t^2 \in F$ (i.e., p_t^2 is an accepting state of M);
4. For every $2 \leq j \leq t$ one has $(p_{j-1}^2, p_j^1) \in E(G)$.

The idea behind the above definition of admissible portals is simple: Given an admissible path A, an admissible sequence P of portals describes how a computation path for a word $w \in L(M)$ moves between the strongly connected components:

- First, it starts from the initial state q_1 with counter 0 and reaches the state p_1^1 with counter l such that (C_{i_1}, p_1^1, l) is non-degenerate (It might be that $A = \emptyset$ in which case all triples are degenerate).
- Then, it moves from one strongly connected component of A to the next one, where every component C_{i_j} is entered at state p_j^1 and exited at state p_j^2, ending in an accepting state $p_t^2 \in C_{i_t}$.

Now, given an admissible path A and a corresponding admissible sequence P of portals, an increasing sequence of integers $\Pi = (n_j)_{j=1}^{t+1}$ forms an *admissible* partition with respect to (A, P) if the following holds:

1. $n_1 \geq 0$;
2. for every $j \in [t]$, there exists a path from p_j^1 to p_j^2 in C_{i_j} of length $n_{j+1} - n_j - 1$;
3. $n_{t+1} = n + 1$.

This partition defines the decomposition of the input word into subwords corresponding to strongly connected components.

We say that l is an *admissible* counter value if (C_{i_1}, p_1^1, l) is non-degenerate and $0 \leq l \leq |V(G)| - 1$. In the full paper we show that if (C_{i_1}, p_1^1, l) was the first non-degenerate triplet in a computation path of w on M then $0 \leq l \leq |V(G)| - 1$.

A 4-tuple (A, P, Π, l) is *admissible*, if A is an admissible path, P is a corresponding admissible sequence of portals, Π is a corresponding admissible partition and l is an admissible counter value.

Definition 8. *[The 4-tuple defined by a computation path] An accepting computation path for $w \in L(M) \cap \{0,1\}^n$ in M defines a 4-tuple (A, P, Π, l), where the sequence $A = (C_{i_1}, \ldots, C_{i_t})$ is such that $C_{i_1}, \ldots, C_{i_t}$ are strongly connected components in the computation path discarding all strongly connected components before the first non-degenerate triplet was encountered, $P = (p_j^1, p_j^2)_{j=1}^t$ such that p_j^1 (resp. p_j^2) is the first (resp. the last) state of C_{i_j} visited by the computation path, $\Pi = (n_j)_{j=1}^{t+1}$ such that $n_1 \geq 0$, $n_{t+1} = n + 1$, and for $1 \leq j \leq t$, n_j is set to be the first index in which w enters C_{i_j} in the computation path, and l is such that (C_{i_1}, p_1^1, l) is a non-degenerate triplet.*

Note that any word $w \in L(M) \cap \{0,1\}^n$ has a computation path, and by a claim proved in the full version, a value for l, for which it defines an admissible 4-tuple. On the other hand, an admissible 4-tuple is enough to convince us that $w \in L(M) \cap \{0,1\}^n$. We stress again that the 4-tuple defined by a computation path can be empty. This occurs when all triplets defined by the computation path are degenerate.

Therefore, in order to be convinced that $w \notin L(M)$, it is enough to check that w does not fit any admissible 4-tuple. Since the number of all admissible 4-tuples is large, we restrict our attention only to the limited number of admissible 4-tuples. These restricted 4-tuples, as well as the notion of a sub-automaton, are introduced in Section 5

5 Restricted 4-Tuples and Sub-automata

5.1 Restricted 4-Tuples

Our intention is to let the testing algorithms try and fit the input word w to one of the admissible 4-tuples. Since there are too many of those, we will construct a restricted subset of them. This restricted subset will serve as an ϵ-net of sorts, for computations over the automaton. The construction is similar to the one in [10] and is based on placing $\mathrm{NumTrIn}$ intervals in $[n]$, each of length $\mathrm{LenTrIn}$. Essentially, the restriction will be that transitions between strongly connected components are only allowed inside these intervals.

Let $\mathrm{NumTrIn} = 200 \cdot 10^{|\mathrm{V}|/\epsilon^{10}} \cdot \phi_{\mathrm{G(M)}} \cdot \frac{1}{\epsilon^4}$, where $\phi_{G(M)}$ is a constant we set later, and depends only on the structure of $G(M)$. We place $\mathrm{NumTrIn}$ transition intervals $\{T_s = [a_s, b_s]\}_{s=1}^{\mathrm{NumTrIn}}$ evenly in $[n]$, where the length of each transition interval T_s is $\mathrm{LenTrIn} = 10 \cdot (\mathrm{k}-1)(\psi_{\mathrm{G(M)}} + \phi_{\mathrm{G(M)}})$.

A 4-tuple (A, P, Π, l), where $\Pi = (n_j)_{j=1}^{t+1}$, is called *restricted* if it is admissible and for every $j \in [t]$ we have $n_j \in T_s$ for some $s \in [\mathrm{NumTrIn}]$. In particular, an empty 4-tuple is restricted.

For the rest of the proof, we will need to "glue together" subwords from different strongly connected components. Towards this we set the value $\mathrm{gap} = \frac{\mathrm{n}}{\mathrm{NumTrIn}} + \mathrm{LenTrIn}$. Intuitively, this value is the "slack" left untested in every strongly connected component so they may be "glued" together later.

We now prove that the number of restricted 4-tuples is independent of n.

Claim 2. *The number of restricted 4-tuples with respect to the weak counter automaton $M = (V, q_1, F, \delta)$ is at most*

$$2^{|V|}|V|^{2|V|+1}(\mathrm{NumTrIn} \cdot \mathrm{LenTrIn})^{|\mathrm{V}|}.$$

Proof. Recall that a restricted 4-tuple is denoted by (A, P, Π, l), where we have $A = (C_{i_1}, \ldots, C_{i_t})$, $P = (p_j^1, p_j^2)_{j=1}^t$, $\Pi = (n_j)_{j=1}^{t+1}$ and $0 \leq l \leq |V| - 1$ such that C_{i_j}-s are strongly connected components in $G(M)$, for all $j \in [t]$ we have $p_j^1, p_j^2 \in V$ and $n_j \in T_s$ for some $s \in [\mathrm{NumTrIn}]$.

Note that the number of strongly connected components is at most $|V|$ and thus $t \leq |V|$. The number of possible choices for A (choose an admissible path in $C(G)$) is at most $2^{|V|}$ as any subset of vertices of $C(G)$ defines at most one path spanning it.

The total number of chosen portals (in P) is at most $2|V|$, therefore there are at most $|V|^{2|V|}$ possible choices for portals. Then for a fixed pair (A, P) there are at most $\mathrm{NumTrIn} \cdot \mathrm{LenTrIn}$ choices for each n_j, where $1 \leq j \leq t$ and $t \leq |V|$.

Finally, there are $|V|$ possible choices for the value l. Hence the number of restricted 4-tuples is upper-bounded by

$$2^{|V|}|V|^{2|V|} \cdot (\mathrm{NumTrIn} \cdot \mathrm{LenTrIn})^{|\mathrm{V}|} \cdot |\mathrm{V}|.$$

5.2 Sub-automata

With an admissible 4-tuple in hand (A, P, Π, l), we will restrict M to sub-automata corresponding to the 4-tuple. A formal definition follows.

Definition 9. [Strongly connected automaton] *The automaton $M' = (V, q_1, F, \delta)$ is strongly connected if M' has a unique accepting state q_{acc} and $G(M')$ is a strongly connected graph.*

For every strongly connected component C_{i_j} in A we construct the corresponding strongly connected automaton M_j as follows.

Constructing a sub-automaton. Let (A, P, Π, l) be an admissible 4-tuple such that $A = (C_{i_1}, \ldots, C_{i_t})$, $P = (p_j^1, p_j^2)_{j=1}^t$, $\Pi = (n_j)_{j=1}^{t+1}$ and $0 \leq l \leq |V(G)| - 1$. For all $j \in [t]$, we define a corresponding automaton M_j corresponding to C_{i_j}: The set of states for M_j is C_{i_j}. The initial state of M_j and its unique accepting state are p_j^1 and p_j^2, respectively. For each $q \in C_{i_j}$, $\alpha \in \{0, 1\}$ and $l' \in \mathbb{N}$, if $(q, l') \rightarrow_\alpha^M (q', l'')$ for some $q' \in C_{i_j}$, we set $(q, l') \rightarrow_\alpha^{M_j} (q', l'')$. Other transitions are not reflected in the automaton M_j. Note that by its definition, M_j is a strongly connected automaton as in Definition 9.

6 Algorithms

In this section we define our testing algorithm. We will use $M = (V, q_1, F, \delta)$ to denote the weak counter automaton. We start by introducing the $\mathrm{Approximator}$ algorithm, which allows us to approximate the behavior of a word inside a strongly connected automaton. Next, we introduce the main testing algorithm which uses the $\mathrm{Approximator}$ and $\mathrm{TestRegularLang}$ (which is defined in the full version of the paper and is essentially a test for membership in a regular language) to test all possible restricted 4-tuples.

We defer the statement of Algorithm $\mathrm{Approximator}$ and its proof of correctness to the full version, and only state the claims regarding its correctness.

The input to $\mathrm{Approximator}$ algorithm is given by $w \in \{0, 1\}^{n'}$, $\epsilon > 0$ and a strongly connected automaton $M' = (V', q_1', F' = \{q_f\}, \delta')$. The algorithm outputs a triplet, where the first entry is "true" or "false", and the other two are numerical values. If the algorithm found no evidence that w is infeasible then the first entry in the triplet is "true", and the next two entries should approximate the minimal prefix and value of w in M'. Else, the first entry in the triplet is "false", and the other two are irrelevant. Lemma 1 states that if w is feasible (up to changes in the gap first and last bits) then $\mathrm{Approximator}$ outputs "true" together with a good estimation of the minimal prefix and value with high probability.

Lemma 1. *[Completeness] Assume that $w \in \{0, 1\}^{n'}$, $\epsilon > 0$ and the automaton $M' = (V', q_1', F' = \{q_f\}, \delta')$ are the input to* $\mathrm{Approximator}$. *Assume that $w[\mathrm{gap}, \mathrm{n}' - \mathrm{gap}]$ is feasible with respect to M'. Then, with probability at least $1 - \frac{1}{100|V|}$ we have that* $\mathrm{Approximator}$ *outputs*

$$(true, minPref, value)$$

such that

- $value - \mathrm{val}_{\mathrm{M}'}(\mathrm{w}[\mathrm{gap}, \mathrm{n}' - \mathrm{gap}]) \geq -\epsilon \mathrm{n}'$, *and*
- $minPref - \mathrm{minPref}_{\mathrm{M}'}(\mathrm{w}[\mathrm{gap}, \mathrm{n}' - \mathrm{gap}]) \geq -\epsilon \mathrm{n}'$.

Lemma 2 states that either Approximator returns "false" with high probability, or there is a word w' close to w which is feasible and its minimal prefix and value are close to those estimated by the algorithm with high probability.

Lemma 2. *[Soundness and robustness] Assume that* $w \in \{0,1\}^{n'}$, $\epsilon > 0$ *and the automaton* $M' = (V', q_1', F' = \{q_f\}, \delta')$ *are the input to* Approximator. *If* w *is* ϵ*-far from* $L(M')$, Approximator *outputs* $(false, \cdot, \cdot)$ *with probability at least* $1 - \frac{\epsilon}{10\mathrm{NumTuples}}$. *If* $w \in L(M')$ *then with probability at least* $1 - \frac{1}{10|V|\mathrm{NumTuples}}$, *the* Approximator *algorithm outputs* $(true, minPref, value)$ *such that there exists* $w' \in \{0,1\}^{n'}$ *such that*

- $|value - \mathrm{val}_{\mathrm{M}'}(\mathrm{w}')| \leq \epsilon n' + 2\mathrm{gap}$,
- $|minPref - \mathrm{minPref}_{\mathrm{M}'}(\mathrm{w}')| \leq \epsilon n' + 2\mathrm{gap}$,
- $(q_1', -minPref + \epsilon n' + 2\mathrm{gap}) \rightarrow_{\mathrm{w}'} (\mathrm{q_f}, \cdot)$, *and*
- $\Delta(w, w') \leq \epsilon n' + 2\mathrm{gap}$.

Claim 3. *The number of queries used by* Approximator *is at most*

$$(p \cdot |Start| \cdot \mathrm{size}) \cdot (1000|\mathrm{V}|) = \mathrm{O}_{\epsilon,|\mathrm{V}|}(1).$$

Moreover, Approximator *is non-adaptive.*

6.1 Testing Algorithm — Algorithm 1

Given a restricted 4-tuple (A, P, Π, l) with $A = (C_{i_1}, \ldots, C_{i_t}), P = (p_j^1, p_j^2)_{j=1}^t$ and $\Pi = (n_j)_{j=1}^{t+1}$ we denote by $A_j(M)$ the strongly connected counter automaton (Definition 9) associated with the j-th strongly connected component C_{i_j} in A. We denote by $\Pi(j)$ the j-th element in Π, and by $P(p_1^1)$ the first state in the first pair in P.

First the testing algorithm checks the empty 4-tuple using TestRegularLang (defined in the full version of the paper). Then the testing algorithm proceeds to iterate over all non-empty restricted 4-tuples. The testing of each non-empty 4-tuple contains two steps. First, we test whether the $(P(p_1^1), l)$ is $(|V|-1)$-reachable from $(q_1, 0)$ on the word $w[1, \Pi(1)]$ (see the full version for details). If this test fails the 4-tuple is rejected. Otherwise we proceed to the next step.

For the next step we partition the input word w into t subwords $w^1, \ldots, w^t$ by setting $w^j = w[\Pi(j) + 1, \Pi(j+1)]$. We estimate the counter change caused by the computation path of $w[\Pi(j) + 1, \Pi(j+1)]$ in $A_j(M)$ using the Approximator (further described in the full version of the paper). If at least one strongly connected component is rejected by the Approximator then this 4-tuple is rejected. If all strongly connected components are not rejected, we then use the approximations it obtains to decide whether the 4-tuple can be accepted.

The correctness of Algorithm 1 is proved in the full version of the paper, implying Theorem 1.

Algorithm 1. Tester for Weak Counter Automata

Input: $M = (V, q_1, F, \delta)$, $w \in \{0,1\}^n$ and $\epsilon > 0$.
Initialize: $\epsilon' = \frac{\epsilon}{100}$, $\epsilon'' = \frac{\epsilon^{10}}{3 \cdot 10^5 |V|^5}$
if $\mathrm{TestRegularLang(M, w[1, n], F \times \{0, 1, \ldots, |V| - 1\}, \epsilon')}$ **then**
  Output accept.
for every non-empty restricted 4-tuple (A, P, Π, l) **do**
  if $\mathrm{TestRegularLang(M, w[1, \Pi(1)], \{(P(p_1^1), l)\}, \epsilon')}$ **then** {Defined in the full version of the paper}
    $value := 0, minPref := 0, valTmp := 0$
    $j := 1$
    $success := true$
    while $(j \leq |A|)$ & $success$ **do**
      $(success, minPref, valTmp) := \mathrm{Approximator(A_j(M), w[\Pi(j) + 1, \Pi(j+1)], \frac{\epsilon''}{2|V|})}$
      if $value + minPref < -\epsilon'' n$ **then**
        $success := false$
      $value := value + valTmp$
      $j := j + 1$
    if success **then**
      Output accept.
Output reject.

Acknowledgements. The authors wish to thank Oded Lachish and Eldar Fischer for their invaluable insights and ideas on the problem.

References

1. Alon, N., Krivelevich, M., Newman, I., Szegedy, M.: Regular languages are testable with a constant number of queries. SIAM J. Comput. 30(6), 1842–1862 (2001), http://dx.doi.org/10.1137/S0097539700366528
2. Ben-Sasson, E., Harsha, P., Raskhodnikova, S.: Some 3CNF properties are hard to test. SIAM J. Comput. 35(1), 1–21 (2005), http://dx.doi.org/10.1137/S0097539704445445
3. Fischer, E.: The art of uninformed decisions: A primer to property testing. In: Current Trends in Theoretical Computer Science: The Challenge of the New Century I, pp. 229–264 (2004)
4. Fischer, E., Goldhirsh, Y., Lachish, O.: Testing formula satisfaction. In: Fomin, F.V., Kaski, P. (eds.) SWAT 2012. LNCS, vol. 7357, pp. 376–387. Springer, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-31155-0
5. Fischer, E., Lehman, E., Newman, I., Raskhodnikova, S., Rubinfeld, R., Samorodnitsky, A.: Monotonicity testing over general poset domains. In: Reif, J.H. (ed.) Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC), Montréal, Québec, Canada, May 19-21, pp. 474–483. ACM (2002), http://doi.acm.org/10.1145/509907.509977
6. Fischer, E., Newman, I., Sgall, J.: Functions that have read-twice constant width branching programs are not necessarily testable. Random Struct. Algorithms 24(2), 175–193 (2004), http://dx.doi.org/10.1002/rsa.10110

7. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. J. ACM 45(4), 653–750 (1998), http://doi.acm.org/10.1145/285055.285060
8. Halevy, S., Lachish, O., Newman, I., Tsur, D.: Testing properties of constraint-graphs. In: IEEE Conference on Computational Complexity, pp. 264–277. IEEE Computer Society (2007), http://doi.ieeecomputersociety.org/10.1109/CCC.2007.31
9. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation, 2nd edn. Addison-Wesley, New York (2001)
10. Alon, N., Fischer, E., Krivelevich, M., Szegedy, M.: Efficient testing of large graphs. Combinatorica (20), 451–476 (2000)
11. Lachish, O., Newman, I., Shapira, A.: Space complexity vs. query complexity. Computational Complexity 17(1), 70–93 (2008), http://dx.doi.org/10.1007/s00037-008-0239-z
12. Newman, I.: Testing membership in languages that have small width branching programs. SIAM Journal on Computing 31(5), 1557–1570 (2002), http://epubs.siam.org/sam-bin/dbq/article/38211
13. Parnas, M., Ron, D., Rubinfeld, R.: Testing membership in parenthesis languages. Random Struct. Algorithms 22(1), 98–138 (2003), http://dx.doi.org/10.1002/rsa.10067
14. Ron, D.: Property testing: A learning theory perspective. Foundations and Trends in Machine Learning 1(3), 307–402 (2008), http://dx.doi.org/10.1561/2200000004
15. Ron, D.: Algorithmic and analysis techniques in property testing. Foundations and Trends in Theoretical Computer Science 5(2), 73–205 (2009), http://dx.doi.org/10.1561/0400000029
16. Rubinfeld, R., Sudan, M.: Testing polynomial functions efficiently and over rational domains. In: Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 23–43 (1992)
17. Rubinfeld, R., Sudan, M.: Robust characterizations of polynomials with applications to program testing. SIAM J. Comput. 25(2), 252–271 (1996), http://dx.doi.org/10.1137/S0097539793255151
18. Valiant, L.G., Paterson, M.: Deterministic one-counter automata. J. Comput. Syst. Sci 10(3), 340–350 (1975), http://dx.doi.org/10.1016/S0022-0000(75)80005-5

Tight Lower Bounds for Testing Linear Isomorphism

Elena Grigorescu[1,⋆], Karl Wimmer[2,⋆⋆], and Ning Xie[3,⋆⋆⋆]

[1] Department of Computer Science, Purdue University, West Lafayette, IN
elena-g@purdue.edu
[2] Department of Mathematics, Duquesne University, Pittsburgh, PA
wimmerk@duq.edu
[3] SCIS, Florida International University, Miami, FL
nxie@cs.fiu.edu

Abstract. We study lower bounds for testing membership in families of linear/affine-invariant Boolean functions over the hypercube. Motivated by the recent resurgence of attention to the permutation isomorphism problem, we first focus on families that are linearly/affinely isomorphic to some fixed function.

Our main result is a tight adaptive, two-sided $\Omega(n^2)$ lower bound for testing linear isomorphism to the inner-product function. This is the first lower bound for testing linear isomorphism to a specific function that matches the trivial upper bound. Our proof exploits the elegant connection between testing and communication complexity discovered by Blais *et al.* (Computational Complexity, 2012.)

Our second result shows an $\Omega(2^{n/4})$ query lower bound for any adaptive, two-sided tester for membership in the Maiorana-McFarland class of bent functions. This class of Boolean functions is also affine-invariant and its rich structure and pseudorandom properties have been well-studied in mathematics, coding theory and cryptography.

1 Introduction

A *property* $\mathcal{P}$ is a set of objects that share some common features. A local test for a property $\mathcal{P}$ is a randomized algorithm which can distinguish inputs that belong to $\mathcal{P}$ from inputs that are very different from every element in $\mathcal{P}$, by making only a few queries to the input. In this work we focus on families of boolean functions $\mathcal{P} \subseteq \{\mathbb{F}_2^n \to \mathbb{F}_2\}$, where $\mathbb{F}_2 = \{0, 1\}$ is the field on two elements. Formally, a (δ, k)-tester for $\mathcal{P}$ is a randomized algorithm that has oracle access to a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, makes k queries, and accepts w.p. at least $2/3$ if $f \in \mathcal{P}$, while rejecting w.p. at least $2/3$ if f is δ-far from $\mathcal{P}$. The notion of distance to a

⋆ Research supported in part by NSF grant 1019343 to the Computing Research Association for the CIFellows Project.
⋆⋆ Supported by NSF award CCF-1117079.
⋆⋆⋆ Part of the work was done when the author was at CSAIL, MIT and was supported by NSF awards CCF-1217423 and CCF-1065125.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 559–574, 2013.

property is given by the relative Hamming distance, namely for $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$, $\text{dist}(f,g) = \frac{1}{2^n}|\{x : f(x) \neq g(x)\}|$ and $\text{dist}(f, \mathcal{P}) = \min_{g \in \mathcal{P}} \text{dist}(f,g)$. If a test always accepts function $f \in \mathcal{P}$ it is called *one-sided*, otherwise it is *two-sided*. If the test must send the queries all at once it is called *non-adaptive*, otherwise, namely when the queries could depend on answers to previous queries, the test is *adaptive*.

The field of property testing was pioneered by Blum, Luby, and Rubinfeld [1], Rubinfeld and Sudan [2] and Goldreich, Goldwasser and Ron [3] who introduced two major directions in property testing: testing algebraic properties and testing combinatorial (e.g. graph) properties. To a large extent, the focus of property testing so far has been to characterize what properties admit testers that make only a constant number of queries (these properties are called *strongly testable*). Alon *et al.* [4] and Borgs *et al.* [5] already showed a complete characterization of strongly testable properties of dense graphs. Very recently, Bhattacharyya *et al.* [6] announced a characterization of one-sided strongly testable boolean families that are invariant under "affine" transformations of the domain.

A systematic study of strongly testable properties that are invariant under natural transformations of the domain was first proposed by Kaufman and Sudan [7]. The most studied and at the same time most natural group of invariances for properties defined over structured, discrete objects such as fields or vector spaces are linear and affine transformations of the domain. A linear transformation $L_C : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is a mapping $L_C(x) = Cx$, where $C \in \mathbb{F}_2^{n \times n}$. An affine transformation $L_{C,b} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is a mapping $L_{C,b}(x) = Cx + b$, where $C \in \mathbb{F}_2^{n \times n}$ and $b \in \mathbb{F}_2^n$. A property $\mathcal{P} \subset \{\mathbb{F}_2^n \to \mathbb{F}_2\}$ is linear-invariant if $f \in \mathcal{P}$ if and only if $f \circ L_C \in \mathcal{P}$, for any linear transformation L_C, where $f \circ L_C(x) = f(L_C(x))$. Similarly, $\mathcal{P}$ is affine-invariant if $f \in \mathcal{P}$ if and only if the function $f \circ L_{C,b} \in \mathcal{P}$, for any affine $L_{C,b}$, where $f \circ L_{C,b}(x) = f(L_{C,b}(x))$. Following [7] linear/affine invariant families have been intensely studied on two fronts: properties that arise in the setting of linear codes [7,8,9,10,11,12,13,14,15], and properties that arise more often in the study of boolean functions [16,17,18,19,6]. All these works study properties that are testable with a constant number of queries.

Here we work in a somewhat complementary direction: we study linear/affine-invariant properties that are hard to test. Partly motivated by the recent resurgence of interest in the permutation isomorphism problem [20,21,22,23,24,25], we focus on testing linear/affine isomorphism to a single function. This study, in a certain sense, combines the directions of testing linear/affine-invariance and testing permutation isomorphism. As isomorphism defines an equivalence relation among functions in the family, we restrict our attention to *non-singular* linear/affine transformations in this paper[1] (non-singular transformations lead to permutations of a function while singular transformations do not). More specifically, the orbit of a function f under the set of non-singular linear transformations of $\mathbb{F}_2^n$ is given by $\mathcal{L}(f) = \{f \circ L_C | C \in \mathbb{F}_2^{n \times n}, \det(C) = 1\}$; a function g is said to be *linearly isomorphic* to f if $g \in \mathcal{L}(f)$. Similarly, the orbit of f under affine

[1] The reason of such a choice is explained later in Remark 1.

transformations is the family $\mathcal{A}(f) = \{f \circ L_{C,b} | C \in \mathbb{F}_2^{n \times n}, \det C = 1, b \in \mathbb{F}_2^n\}$ and g is *affinely isomorphic* to f if $g \in \mathcal{A}(f)$. For instance, when $f = x_1$ is a dictator function, $\mathcal{L}(f)$ is just the set of non-constant, linear functions, and $\mathcal{A}(f)$ is the set of non-constant, affine functions.

We exhibit a large family of functions for which testing linear/affine isomorphism to every function in the family requires $\Theta(n^2)$ many queries. Our explicit functions arise from families of Boolean *bent* functions. Bent functions are the functions that are the most 'uncorrelated' with linear functions (see Section 2 for the precise definition that we use). A most common example of bent functions, which is also an object of interest in this work, is the inner-product function $\mathrm{IP}_n : \mathbb{F}_2^n \to \mathbb{F}_2$ defined by $\mathrm{IP}_n(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n/2} x_{2i-1}x_{2i}$ (here and in what follows we will assume that n is a multiple of 4).

Bent functions have been well-investigated in mathematics, coding theory and combinatorial design [26,27,28,29,30] for their rich structure, and in differential cryptography [31,32] for their pseudorandom and non-linearity properties that make them applicable to building hash-functions; see e.g. [33] for a comprehensive survey. In property testing they were used before in [34] to show lower bounds for testing triangle freeness.

In this work we also show exponential lower bounds for testing membership in the class of bent functions, which is invariant under non-singular affine transformations of the domain. Hence, our results reveal yet some novel uses of bent functions in property testing, suggesting that they have some inherent feature that make them hard for local testing algorithms.

1.1 Our Results

Lower bounds for testing linear/affine isomorphisms. We start with the family $\mathcal{P} = \mathcal{L}(\mathrm{IP}_n)$ that has size $|\mathcal{P}| = 2^{O(n^2)}$. We show that the query complexity of the trivial algorithm (which simply picks $O(n^2)$ random inputs and checks if there is a function in $\mathcal{P}$ that agrees with the answers to the queries) is in fact asymptotically optimal, even for *adaptive, 2-sided tests.* Since the query complexity of the single-sided, non-adaptive tester is no less than that of the 2-sided, adaptive tester, this result shows that $\mathcal{L}(\mathrm{IP})$ is an example of a family that is the hardest to test for linear isomorphism.

Theorem 1. *Any 2-sided, adaptive $(1/4, k)$-test for $\mathcal{L}(\mathrm{IP}_n)$ and $\mathcal{A}(\mathrm{IP}_n)$ requires $k = \Omega(n^2)$ queries.*

We remark that a lower bound of $\Omega(n)$ follows from the results of Chakraborty *et al.* [22], since the functions in these families have Fourier dimension n, and can be shown to be far from having (Fourier) dimension $n - 1$.

Remark 1. Note that since $\mathcal{P} = \mathcal{L}(\mathrm{IP}_n)$ is a collection of polynomials of degree 2, it is a subset of Reed-Muller codes of order 2, RM(2). However, using Dickson's theorem, one can show that, if $\mathcal{L}$ is not restricted to non-singular linear transformations, then the set of functions $\mathcal{L}(\mathrm{IP}_n)$ is identical to RM(2).

The latter is well-known to be testable with 8 queries by a single-sided non-adaptive tester. What we show here is that testing a subset of RM(2), $\mathcal{L}(\mathrm{IP}_n)$, is much harder.

We generalize this result to a broader class of bent functions commonly known as Maiorana-McFarland bent functions, denoted $\mathcal{MM}_n$. Formally, a function in this family is $f : \mathbb{F}_2^n \to \mathbb{F}_2$ defined by $f(x, y) = \langle x, y \rangle + g(y)$, where $x, y \in \mathbb{F}_2^{n/2}$ and $g : \mathbb{F}_2^{n/2} \to \mathbb{F}_2$ is arbitrary, and $\langle x, y \rangle$ denotes the inner product (standard dot product) of x and y.

Notice that these functions are no longer low-degree polynomials, and in fact, since g is arbitrary they could be polynomials of degrees as high as $n/2$. For these families too, we show that testing isomorphism is hard.

Theorem 2. *Any 2-sided, adaptive $(1/4, k)$-test for $\mathcal{L}(f)$ and $\mathcal{A}(f)$ where f is a Maiorana-McFarland bent function in n variables requires $k = \Omega(n^2)$ queries.*

We remark that the arguments of Alon and Blais from [21] for testing isomorphism under the symmetric group can be easily adapted to testing isomorphism under the group of non-singular transformations to show that testing linear isomorphism to almost all functions requires $\Omega(n)$ queries.

To the best of our knowledge, no superlinear (in n) 2-sided error lower bounds have been previously established for testing any *explicitly given* function under some class of isomorphisms. More generally, let $f : D \to \mathbb{F}_2$ be a function, and G be a group acting on D, where D is some finite domain; (informally, the elements of the group can be identified with bijections from D to itself). The original question of (permutation) isomorphism considers $D = \mathbb{F}_2^n$ and $G = \mathbb{S}_n$, the symmetric group, acting on D in the natural way. Linear isomorphism, which is the focus of this paper, has $D = \mathbb{F}_2^n$ and $G = GL(n, \mathbb{F}_2)$, the group of invertible matrices over $\mathbb{F}_2$, acting on D in the natural way. The easy upper bound for a testing isomorphism algorithm is $\log |G|$, but for technical reasons, many recent lower bounds and their analyses get bottlenecked by $\log |D|$. For permutation isomorphism, this gap is still open. For linear isomorphism, we close this gap in this paper.

In the case that $D = G$ and the group action is simply the group operation, we remark that the proof of [21] can be easily modified to yield the following:

Theorem 3. *Let G be a finite group, and choose a random function $f : G \to \{0, 1\}$. Testing isomorphism to f under the group action of multiplying by an element of G requires $\Omega(\log |G|)$ queries with high probability.*

We omit the details of this proof. In this case, almost every function requires as many queries for testing isomorphism as the most simple algorithm.

Lower bounds for testing bentness and Maiorana-McFarland families. We then turn to analyzing bent functions in general. The number of boolean bent functions is known to be at least $2^{2^{n/2+\Omega(\log n)}}$ and the current upper bound is larger than $2^{2^{n-1}}$ [35]. We show a lower bound of $\Omega(2^{n/4})$ queries for testing bentness.

A lower bound of significantly more than $2^{n/2+\Omega(\log n)}$ queries for the testing problem would improve the current status regarding the number of boolean bent functions, a long standing open problem in mathematics. We note that no non-trivial test (that makes substantially fewer than 2^n queries) for bentness is currently known.

Theorem 4. *Any 2-sided, adaptive $(1/4, k)$-test for the class of boolean bent functions in n variables requires $k = \Omega(2^{n/4})$ queries.*

Theorem 4 is in fact an immediate consequence of a same lower bound for testing the Maiorana-McFarland family. Let $\mathcal{L}(\mathcal{MM}_n)$ be the linear closure of the family $\mathcal{MM}_n$, i.e. $\mathcal{L}(\mathcal{MM}_n) = \bigcup_{f \in \mathcal{MM}_n} \mathcal{L}(f)$.

Theorem 5. *Any 2-sided, adaptive $(1/4, k)$-test for testing membership in $\mathcal{L}(\mathcal{MM}_n)$ requires $k = \Omega(2^{n/4})$ queries.*

We remark that exponential lower bounds for testing affine-invariant properties were known before. For example, testing the Reed-Muller code of degree $n/2$ requires at least $2^{n/2} - 1$ queries, as $2^{n/2}$ is the minimum distance of its dual code.[2] Hence this lower bound is interesting mainly in the context of testing bent functions and their generalization Maiorana-McFarland families. There results appear in section 4.

Finally, we remark that all our results generalize to non-boolean functions over prime fields $\mathbb{F}_p$, but we defer the proofs to the full version of this paper.

1.2 Previous Related Work

Recently there has been intense interest in testing isomorphism of functions (with respect to the symmetric group). Some initial results of this type can be attributed to [36] who showed the strong testability of dictators and monomials. The problem of testing isomorphism was first explicitly introduced by [37]. In recent years the interest in testing function isomorphism has revived [20,21,22,23,24,25] prompted by the works of Blais and O'Donnell [20] and Alon and Blais [21]. The main motivation of many of these works (including the original motivation of [37]) involves testing isomorphism to functions with few relevant variables.

Testing linear isomorphism has been less studied. It has been considered by Chakraborty *et al.* [22] who show a lower bound of $\Omega(k)$ for testing $\mathcal{L}(f)$ for a function f that is far from having (Fourier) dimension $k-1$. In line with testing juntas, a previous result of [38] implicitly proves an upper bound of $O(k2^k)$ for linear isomorphism to functions that are very close to having dimension k; the "very" here is exponentially small in k. Wimmer and Yoshida [39] give an constant-query algorithm for linear isomorphism to any function close to having dimension k by giving a tolerant tester for functions of dimension k. The technique is an extension of the work of [38], and it applies to functions close

[2] We thank an anonymous referee for pointing this out.

to having low spectral norm. They also show lower bounds for testing linear isomorphism, but these lower bounds are no better than $\Omega(n)$ for any fixed function.

1.3 Our Techniques

Testing linear/affine isomorphisms to IP*.* Our lower bounds for testing linear/affine isomorphism are proved using reductions from communication complexity protocols, a powerful technique introduced by Blais *et al.* [40]. In the communication complexity model there are two parties holding inputs x and y, respectively, who are trying to compute a function $f(x, y)$ with as little communication between them as possible. In [40], the authors show an ingenious generic technique to prove lower bounds for property testing, by exploiting the strength of the lower bounds obtained in communication complexity.

The crux of our argument is the observation that one can reduce testing linear isomorphism to IP (and more generally, to any Maiorana-McFarland bent function) from the following natural randomized communication protocol: Alice holds the top half of a matrix C, Bob holds the remaining half of C, and their goal is to determine if C is singular. The main feature of bent functions that we make use of here is the fact that when composed with singular linear transformations they not only become functions that are not bent, but they become functions that are *far* from bent (See Proposition 1). To complete the proof we resort to the recent results of Sun and Wang [41] who show a lower bound of $\Omega(n^2)$ for the randomized communication complexity for this problem.

Testing bentness. To show the lower bound for testing bentness we use Yao's principle, where the Yes distribution is the (linear closure of) Maiorana-McFarland family of functions and the No distribution is supported on random $n/2$ dimensional functions. Our argument that these two distributions are indistinguishable resembles the work of [38]. We show that, for any fixed set Q of $\Omega(2^{n/4})$ queries and for most $(n/2)$-dimensional subspaces H, every vector in Q is in a distinct coset of H. This fact is the statement one would expect given the famous "birthday paradox". Our results shows there are at least $\Omega(2^{n/4})$ "degrees of freedom" in selecting a linear transformation of a Maiorana-McFarland function. This lower bound translates upward to the class of all bent functions, since every bent function is far from the class of $(n/2)$-dimensional functions.

2 Preliminaries

Let $n \geq 1$ be a natural number. We use $[n]$ to denote the set $\{1, \ldots, n\}$. $\mathbb{F}_2 = \{0, 1\}$ is the field with 2 elements, where addition and multiplication are performed mod 2. We view elements in $\mathbb{F}_2^n$ as n-bit binary strings – that is elements of $\{0, 1\}^n$ – alternatively. If x and y are two n-bit strings, then $x + y$ (or $x - y$) denotes bitwise addition (i.e. XOR) of x and y. We view $\mathbb{F}_2^n$ as a vector space equipped with an inner product $\langle x, y \rangle$, which we take to be the standard dot product: $\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$, where all operations are performed in $\mathbb{F}_2$.

We start by recalling a basic fact that we have referred to earlier.

Theorem 6 (folklore). *Let $\mathcal{P} \subseteq \{f : D \to R\}$ (for some finite domain D and range R) be a property of size $|\mathcal{P}|$. Then there is a one-sided error testing algorithm which tests $\mathcal{P}$ with distance parameter ϵ using $O(\frac{1}{\epsilon}\log|\mathcal{P}|)$ queries.*

Linear/affine isomorphism. We say that two functions $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ are *linearly isomorphic* (or *linear isomorphic*) if there exists a non-singular linear transformation $C : \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that $g(x) = f(Cx)$ for all $x \in \mathbb{F}_2^n$. Equivalently, g is linearly isomorphic to f if and only if there exist n linearly independent linear functions $\ell_i(x) : \mathbb{F}_2^n \to \mathbb{F}_2$ such that $g(x) = f(\ell_1(x), \ell_2(x), \ldots, \ell_n(x))$ (a linear function, is a function of the form $\ell(x) = \sum_{j\in[n]} l_j x_j$, where $l_j \in \mathbb{F}_2$). Similarly, two functions $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ are *affinely isomorphic* (or *affine isomorphic*) if there exists a non-singular linear transformation $C : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and a vector $b \in \mathbb{F}_2^n$ such that $g(x) = f(Cx + b)$ for all $x \in \mathbb{F}_2^n$. We define $\mathcal{L}(f)$ to be the set of functions linearly isomorphic to f: $\mathcal{L}(f) = \{f \circ L_C | C \in \mathbb{F}_2^{n\times n}, \det(C) = 1\}$. Similarly, we define $\mathcal{A}$ to be the set of functions affinely isomorphic to f: $\mathcal{A}(f) = \{f \circ L_{C,b} | C \in \mathbb{F}_2^{n\times n}, \det C = 1 \text{ and } b \in \mathbb{F}_2^n\}$.

Bent functions. There are many equivalent definitions of bent functions, for example as functions farthest away from any affine functions, or functions whose Fourier coefficients have the same magnitude. Here we will use another standard definition, due to Rothaus [27].

Definition 1. *A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is bent if for every nonzero vector $h \in \mathbb{F}_2^n$, we have $\Pr_{\boldsymbol{x}}[f(\boldsymbol{x}) = f(\boldsymbol{x} + h)] = 1/2$.*

Given a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, we define $\mathrm{Inv}(f) := \{h : f(x) = f(x + h) \text{ for all } x\}$. The set $\mathrm{Inv}(f)$ forms a subspace of $\mathbb{F}_2^n$, and we define the *dimension* of f to be the codimension of $\mathrm{Inv}(f)$. We use $\dim(f)$ to denote the dimension of f. If $\dim(f) \le k$, we say that f is k-dimensional. This notion of dimensionality is equivalent to the notion of Fourier dimension used in [38]. From their definition it immediately follows that the dimension of any bent function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is n.

The following proposition will be of great importance to us.

Proposition 1. *Suppose $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is a bent function and $\dim(g) < n$. Then $\Pr_{\boldsymbol{x}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] \ge 1/4$.*

Proof. Since $\dim(g) < n$, we have $\mathrm{Inv}(g) \neq \{0\}$, so there exists a nonzero vector $h \in \mathbb{F}_2^n$ such that $g(x) = g(x + h)$ for all $x \in \mathbb{F}_2^n$. From Proposition 1, we know that $\Pr_{\boldsymbol{x}}[f(\boldsymbol{x}) \neq f(\boldsymbol{x} + h)] = 1/2$. We have

$$\begin{aligned}
\Pr_{\boldsymbol{x}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] &= \frac{1}{2}(\Pr_{\boldsymbol{x}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] + \Pr_{\boldsymbol{x}}[f(\boldsymbol{x} + h) \neq g(\boldsymbol{x} + h)]) \\
&= \frac{1}{2}(\Pr_{\boldsymbol{x}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] + \Pr_{\boldsymbol{x}}[f(\boldsymbol{x} + h) \neq g(\boldsymbol{x})]) \\
&\ge \frac{1}{2}(\Pr_{\boldsymbol{x}}[f(\boldsymbol{x}) \neq f(\boldsymbol{x} + h)] = 1/4.
\end{aligned}$$

One well-known class of bent functions are the *Maiorana-McFarland* functions defined as follows. Let $x \in \mathbb{F}_2^{n/2}$, $y \in \mathbb{F}_2^{n/2}$, and let $g : \mathbb{F}_2^{n/2} \to \mathbb{F}_2$ be any Boolean function on $n/2$ variables. Then $\mathrm{MM}_n^g : \mathbb{F}_2^n \to \mathbb{F}_2$ given by $\mathrm{MM}_n^g(x, y) = \langle x, y \rangle + g(y)$ is called a *Maiorana-McFarland function.*

As mentioned before, a formula for the number of bent functions on n variables is unknown. It is worth noting that there are $2^{2^{n/2}}$ distinct bent Maiorana-McFarland functions, so this class accounts for a significant portion of the bent functions known.

We next list a few basic facts about bent functions that will be useful to us later. Due to space constraints, we leave the proof of these facts to the full version of this paper.

Lemma 1

1. *The inner product function* IP_n *is bent.*
2. *Any Maiorana-McFarland function* MM_n^g *is bent.*
3. *If* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *is bent,* $C \in \mathbb{F}_2^{n \times n}$ *is non-singular and* $b \in \mathbb{F}_2^n$*, then* $f \circ L_C$ *and* $f \circ L_{C,b}$ *are bent.*
4. *If* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *is bent,* $C \in \mathbb{F}_2^{n \times n}$ *is singular and* $b \in \mathbb{F}_2^n$*, then* $f \circ L_C$ *and* $f \circ L_{C,b}$ *are* $1/4$*-far from bent.*

Basic communication complexity facts. In communication complexity Alice holds an input x and Bob holds an input y and they want to compute a function $f(x, y)$ by exchanging a small number of bit messages. A randomized protocol with ϵ error for computing f is an algorithm whose random bits are known to both the players, and which outputs $f(x, y)$ for any x, y w.p. at least $1 - \epsilon$ over the choice of random bits. In this paper we will fix $\epsilon = 1/3$. The complexity of the protocol is the maximum over all x, y of the number of bits exchanged by Alice and Bob. The number of random bits used in the protocol does not affect the complexity measure of the protocol. For a comprehensive survey on communication complexity, see [42].

3 Lower Bounds for Testing Isomorphism to Inner-Product and Related Functions

We prove Theorem 1 and Theorem 2 in this section.

As previously mentioned, the main idea of our proofs is an adaption of a technique of proving property testing query lower bounds via communication lower bounds (See Lemma 2.4 of [40]) to the setting of testing linear isomorphisms. Specifically, Alice and Bob are each given half of a linear transformation matrix C, and their goal is to determine if C is singular or not. They can apply their halves, A and B respectively, of matrix C to an arbitrary input x to the inner product function $\mathrm{IP}_n(x)$. Using the fact that $\mathrm{IP}_n(x_1, x_2, \ldots, x_n) = \mathrm{IP}_{n/2}(x_1, \ldots, x_{n/2}) + \mathrm{IP}_{n/2}(x_{n/2+1}, \ldots, x_n)$, Alice computes $\mathrm{IP}_{n/2}(Ax)$, Bob computes $\mathrm{IP}_{n/2}(Bx)$, and both exchange their bits. Now Alice and Bob both know $\mathrm{IP}_n(Cx) = \mathrm{IP}_n(Ax, Bx) = \mathrm{IP}_{n/2}(Ax) + \mathrm{IP}_{n/2}(Bx)$.

Clearly, if the matrix C has full rank, then $\mathrm{IP}_n(Cx) \in \mathcal{L}(\mathrm{IP}_n)$; on the other hand, one can show that if C does not have full rank, then $\mathrm{IP}_n(Cx)$ is far from $\mathcal{L}(\mathrm{IP}_n)$. Therefore, if there is a tester for linear isomorphism of IP_n with q queries, then one can turn such a tester into a communication protocol for Alice and Bob to determine if C has full rank or not, using at most $2q$ bits of communication. The lower bound of Sun and Wang [41] implies that $2q = \Omega(n^2)$, finishing the proof. We formally state their lower bound here.

Theorem 7 (Theorem 3, [41]). *The randomized communication complexity of computing* $\det(A+B)$*, where Alice holds the matrix* $A \in \mathbb{F}_2^{n\times n}$ *and Bob holds the matrix* $B \in \mathbb{F}_2^{n\times n}$ *is* $\Omega(n^2)$*.*

We will use two corollaries of this result, the first of which is implicit in [41].

Corollary 1. *Let* A *and* B *be matrices in* $\mathbb{F}_2^{n/2\times n}$ *such that the last* $n/2$ *columns form a basis for* $\mathbb{F}_2^{n/2}$*. Let* C *be the matrix* $\begin{bmatrix} A \\ B \end{bmatrix}$*. The randomized communication complexity of computing* $\det(C)$ *where* A *is held by Alice and* B *is held by Bob is* $\Omega(n^2)$*.*

Proof. By assumption, both Alice and Bob can reduce A and B to $A' = [A'', I_{n/2\times n/2}]$ and $B' = [B'', I_{n/2\times n/2}]$, where $I_{n/2\times n/2}$ is the identity matrix. Then it can be checked that

$$\det(C) = \det\left(\begin{bmatrix} A \\ B \end{bmatrix}\right) = \det\left(\begin{bmatrix} A' \\ B' \end{bmatrix}\right) = \det\left(A'' + B''\right),$$

which needs $\Omega(n^2)$ bits of communication by Theorem 7.

Corollary 2. *Let* $C \in \mathbb{F}_2^{n\times n}$ *be that* $C = \begin{bmatrix} A_{n/4\times n/2} & 0_{n/4\times n/2} \\ B_{n/4\times n/2} & 0_{n/4\times n/2} \\ 0_{n/2\times n/2} & I_{n/2\times n/2}. \end{bmatrix}$*, where Alice holds* A*, Bob holds* B *and matrices* A *and* B *are under the same assumptions as in Corollary 1. Then the randomized communication complexity of computing* $\det(C)$ *is* $\Omega(n^2)$*.*

Proof. The statement follows from Corollary 1 by noticing that $\det(C) = \det\left(\begin{bmatrix} A \\ B \end{bmatrix}\right)$.

As mentioned before, it will be convenient to think of the rows of a linear transformation $C \in \mathbb{F}_2^{n\times n}$ as a sequence of linear maps $\ell_1, \ell_2, \ldots, \ell_n : \mathbb{F}_2^n \to \mathbb{F}_2$, where $\ell_i(x) = \sum_{j=1}^n a_{ij}x_j$, and so $\mathrm{IP}_n(Cx) = \sum_{i=1}^{n/2} \ell_{2i-1}(x)\ell_{2i}(x)$. Similarly, an affine transformation (C, b) of IP_n can be represented by $\mathrm{IP}_n(Cx + b) = \sum_{i=1}^{n/2}(\ell_{2i-1}(x) + b_{2i-1})(\ell_{2i}(x) + b_{2i})$.

3.1 Proof of Theorem 1

We are now ready to complete the proof of Theorem 1 by formalizing the reduction to testing $\mathcal{L}(\mathrm{IP}_n)$ ($\mathcal{A}(\mathrm{IP}_n)$, respectively) from computing $\det(A+B)$ as in Theorem 7.

Lemma 2. *Suppose there exists a 2-sided, adaptive $(1/4,k)$-test for $\mathcal{L}(\mathrm{IP}_n)$ (or $\mathcal{A}(\mathrm{IP}_n)$, respectively), then there exists a randomized communication protocol with public coins, error $1/3$, and communication complexity $O(k)$ for computing $\det(C)$, where Alice holds the top half $A \in \mathbb{F}_2^{n/2\times n}$ of C and Bob holds the bottom half $B \in \mathbb{F}_2^{n/2\times n}$ of C.*

Proof. We will show the reduction to $\mathcal{L}(\mathrm{IP}_n)$ only, as the reduction to $\mathcal{A}(\mathrm{IP}_n)$ follows from a very similar argument. Let $C = \begin{bmatrix} A \\ B \end{bmatrix}$ and let $\mathcal{T}$ be a $(1/4,k)$-tester for $\mathcal{L}(\mathrm{IP}_n)$. We will use it to construct a communication protocol for $\det(C)$. Let $\ell_1,\ldots,\ell_{n/2}$ be the linear forms representing the rows of A and $\ell_{n/2+1},\ldots,\ell_n$ be the linear forms corresponding to the rows of B. Let $f(x) = \mathrm{IP}_n(Cx) = \sum_{i=1}^{n/2} \ell_{2i-1}(x)\ell_{2i}(x)$.

Claim. If $\det(C)=0$ then $f(x)=\mathrm{IP}_n(Cx)$ is $1/4$-far from $\mathcal{L}(\mathrm{IP}_n)$.

Proof. By Items 1 and 3 of Lemma 1, every function in $\mathcal{L}(\mathrm{IP}_n)$ is bent. By Item 4 of Lemma 1 $f(x)=\mathrm{IP}_n(Cx)$ is $1/4$-far from $\mathcal{L}(\mathrm{IP}_n)$.

In other words, if $\det(C)=1$ then $f\in\mathcal{L}(\mathrm{IP}_n)$; and if $\det(C)=0$ then f is $1/4$-far from $\mathcal{L}(\mathrm{IP}_n)$.

Let $q_1,q_2,\ldots,q_k$ be the set of (possibly adaptive) queries performed by the tester $\mathcal{T}$ on input f. The protocol for Alice and Bob is to communicate in k rounds. Since Alice and Bob have access to unlimited shared randomness and they exchange bits after generating each query q_i, we can assume that both of them know q_{i+1} given $q_1,q_2,\ldots,q_i$ and $f(q_1),f(q_2),\ldots,f(q_i)$. (Initially, Alice and Bob both know q_1.)

We claim that the following protocol computes $\det(C)$ with probability at least $2/3$. Alice computes Aq_1, namely $\ell_1(q_1),\ldots,\ell_{n/2}(q_1)$ and sends to Bob $\sum_{i=1}^{n/4}\ell_{2i-1}(q_1)\ell_{2i}(q_1)$. Bob computes Bq_1 and also $\sum_{i=n/4+1}^{n/2}\ell_{2i-1}(q_1)\ell_{2i}(q_1)$. Using Alice's bit he can now simulate the query $f(q_1)$ by now computing $\mathrm{IP}_n(Cq_1)=\sum_{i=1}^{n/2}\ell_{2i-1}(q_1)\ell_{2i}(q_1)$, and Bob can send $f(q_1)$ to Alice. By repeating this protocol for the remaining queries $q_2,\ldots,q_k$, Bob can finally output the answer that the tester $\mathcal{T}$ would output on f. Since $\mathcal{T}$ succeeds w.p. at least $2/3$ on f, it follows that the protocol correctly computes $\det(C)$ w.p. at least $2/3$. Notice that the total number of bits exchanged is $O(k)$.

Proof (Proof of Theorem 1). Suppose $\mathcal{T}$ is a 2-sided, adaptive $(1/4,k)$-test for $\mathcal{L}(\mathrm{IP}_n)$ (or $\mathcal{A}(\mathrm{IP}_n)$ respectively). Then, in particular, it should distinguish functions $f=\mathrm{IP}_n(Cx)$ (here C is a matrix $C=\begin{bmatrix} A \\ B \end{bmatrix}$ with $A,B\in\mathbb{F}_2^{n/2\times n}$ such that

their last $n/2$ columns form a basis for $\mathbb{F}_2^{n/2}$) from functions that are 1/4-far from $\mathcal{L}(\mathrm{IP}_n)$. By Lemma 2, there exists a communication protocol computing $\det(C)$ of complexity $O(k)$. Finally, by Corollary 1 it must be that $k = \Omega(n^2)$.

3.2 Proof of Theorem 2

The reduction from Lemma 2 can be tweaked to work for the much more general class of Maiorana-McFarland bent functions. Recall that every function in the Maiorana-McFarland $\mathcal{MM}_n$ family of bent functions can be expressed as $\mathrm{MM}_n^g(x) = \sum_{i=1}^{n/2} x_i x_{i+n/2} + g(x_{n/2+1}, \ldots, x_n)$ for some $g : \mathbb{F}_2^{n/2} \to \mathbb{F}_2$, and so $\mathcal{L}(\mathrm{MM}_n^g) = \{\mathrm{MM}_n^g(Ax) |\ A \in \mathbb{F}^{n\times n}, \det(A) = 1\}$.

Our reduction in the previous section can not be directly used, since Alice would have half of the (linear functions acting as) inputs to g and Bob would have the other half. Thus, answering queries might require more than constant communication, degrading the lower bound. In this case, we reduce from a scenario where Alice and Bob will both always know the inputs to g; this preserves the lower bound of $\Omega(n^2)$. The reduction now uses matrices of the special form described in Corollary 2.

We note that $\mathrm{MM}_n^0 = \mathrm{IP}_n$, where 0 is the constant 0 function, but our previous reduction to inner product is not equivalent to the following reduction setting $g = 0$.

Lemma 3. *Suppose there exists a 2-sided, adaptive $(1/4, k)$-test for $\mathcal{L}(f)$ (or $\mathcal{A}(f)$, respectively), where $f \in \mathcal{MM}_n$. Then there exists a randomized communication protocol with public coins, error at most 1/3 and communication complexity $O(k)$ for computing* $\det\left(\begin{bmatrix} A_{n/4\times n/2} & 0_{n/4\times n/2} \\ B_{n/4\times n/2} & 0_{n/4\times n/2} \\ 0_{n/2\times n/2} & I_{n/2\times n/2}. \end{bmatrix}\right)$*, where Alice holds A and Bob holds B.*

Proof. Once again, we will only show the proof assuming the existence of a tester for $\mathcal{L}$, since the remaining part of the proof follows by similar arguments.

Let $\mathcal{T}$ be a $(1/4, k)$-tester for $\mathcal{L}(\mathrm{MM}_n^g)$, where $\mathrm{MM}_n^g \in \mathcal{M}$. Let f be the function defined by $f(x) = \mathrm{MM}_n^g(Cx)$. Alice and Bob can simulate queries to the function f as before. Let $q_1, \ldots, q_k$ be a set of (possibly adaptive) queries that $\mathcal{T}$ would make on f. As before, the protocol runs in k rounds, Alice and Bob have unlimited shared randomness, and we may assume that both of them know q_{i+1} given $q_1, \ldots, q_i$ and $f(q_1), f(q_2), \ldots, f(q_i)$.

As before, we view the rows of C as linear transformations $\ell_1, \ell_2, \ldots, \ell_n$. The last $n/2$ rows of C are known to both Alice and Bob, so each of them know $\ell_{n/2+1}, \ldots, \ell_n$. Each of these transformation is a projection on to a single coordinate.

Alice computes $[A\ \ 0]q_1$, namely $\ell_1(q_1), \ldots, \ell_{n/4}(q_1)$ and then she sends to Bob $\sum_{i=1}^{n/4} \ell_i(q_1)\ell_{i+n/2}(q_1)$. Bob computes $[B\ \ 0]q_1$, namely $\ell_{n/4+1}(q_1) \ldots, \ell_{n/2}(q_1)$, and uses the result to compute $\sum_{i=n/4+1}^{n/2} \ell_i(q_1)\ell_{i+n/2}(q_1)$. Now Bob can simulate the query $f(q_1)$ by computing

$$\mathrm{MM}_n^g(Cq_1) = \sum_{i=1}^{n/2} \ell_i(q_1)\ell_{i+n/2}(q_1) + g(\ell_{n/2+1}(q_1), \ell_{n/2+2}(q_1), \ldots, \ell_{n-1}(q_1), \ell_n(q_1)).$$

He can then send this bit back to Alice. After simulating all the queries Bob can output the output of $\mathcal{T}$ on f when the test performed these queries. If $\det(C) = 1$ then $f \in \mathcal{L}(\mathrm{MM}_n^g)$ and the test accepts w.p. at least 2/3, and otherwise, by Lemma 1 f is 1/4-far from $\mathcal{L}(\mathrm{MM}_n^g)$ and the test rejects w.p. at least 2/3. Therefore, the communication protocol succeeds w.p. at least 2/3.

The proof of Theorem 2 follows by a similar argument as in the proof of Theorem 1 but where now we use Lemma 3 and Corollary 2 instead.

4 Testing Linear Isomorphism to the Class of Maiorana-McFarland Bent Functions

Our lower bounds will be established via Yao's minimax principle. We denote the total variation distance between two distributions D_1 and D_2 as $\|D_1 - D_2\|_{TV} := \frac{1}{2}\sum_x |D_1(x) - D_2(x)|$, and our goal is to show that the query responses over a "yes" distribution and a "no" distribution are close in total variation distance.

We define D_{YES} to be the uniform distribution over $\mathcal{L}(\mathcal{MM}_n)$, and D_{NO} to be the uniform distribution over $(n/2)$-dimensional functions. We remind the reader that every function in the support of D_{YES} is (1/4)-far from every function in D_{NO}. In fact, since by Proposition 1 every bent function is (1/4)-far from every function in D_{NO}, this same argument establishes a lower bound for testing bentness[3], and thus prove Theorem 4.

We can simulate random draws from D_{YES} and D_{NO} in the following way. In both experiments, we pick a random function $g : \mathbb{F}_2^n \to \mathbb{F}_2$ and a random full rank $n/2 \times n$ matrix A_b. A draw $f \sim D_{\mathrm{NO}}$ is the function defined by $f(x) = g(A_b x)$, where a draw $f \sim D_{\mathrm{YES}}$ is the function defined by $f(x) = IP_n(\begin{bmatrix} A_t \\ A_b \end{bmatrix} x) + g(A_b x)$, where A_t is a random full rank $n/2 \times n$ matrix such that $\begin{bmatrix} A_t \\ A_b \end{bmatrix}$ is nonsingular (chosen dependently on A_b, the distribution on A_t given A_b will not matter as long as the nonsingularity condition is satisfied). Our approach here is very reminiscent of the approach in [38] for showing a lower bound for testing functions of Fourier dimension k.

In the following, when we refer to a random matrix in $\mathbb{F}_2^{n/2\times n}$, we mean a matrix whose entries are chosen to be 0 or 1 independently and uniformly at random.

Lemma 4. *Let A_b be a random matrix in $\mathbb{F}_2^{n/2\times n}$. Then A_b is full rank (in the $\mathbb{F}_2$ sense) except with probability at most $(n/2)2^{-n/2}$.*

[3] The 1/4 can be replaced by any positive constant less than 1/2; we omit the details.

Proof. The probability that A_b is full rank is $\prod_{i=1}^{n/2}(1-2^{i-n}) \leq (1-2^{-n/2})^{n/2} \leq 1-(n/2)2^{-n/2}$.

Because this probability is subconstant, for the sake of conciseness we will treat this event as always occurring. From now on, we will assume A_b has rank $n/2$ with certainty.

Lemma 5. *Let q_1 and q_2 be two distinct vectors in $\mathbb{F}_2^n$, and A_b be a random matrix in $\mathbb{F}_2^{n/2\times n}$. Then $\Pr_{A_b}[A_bq_1 = A_bq_2] = 2^{-n/2}$.*

Proof. The event is equivalent to $A_b(q_1 - q_2) = 0$, where $q_1 - q_2$ is a fixed nonzero vector. Since $q_1 - q_2$ is nonzero and the rows of A_b are chosen independently and uniformly from $\mathbb{F}_2^n$, the distribution over $A_b(q_1 - q_2)$ is uniform. Thus, $\Pr_{A_b}[A_b(q_1 - q_2) = 0] = \Pr_{A_b}[A_bq_1 = A_bq_2] = 2^{-n/2}$.

Lemma 6. *Let Q be a set of $k = 2^{n/4}/10$ vectors in $\mathbb{F}_2^n$. Let A_b be a random $\{0,1\}$ matrix of dimensions $n/2 \times n$. Then $\Pr_{A_b}[\exists q_1, q_2 \in Q$ such that $A_bq_1 = A_bq_2] \leq 1/100$.*

Proof. We use the previous lemma and the union bound. There are at most $\binom{k}{2} \leq k^2 = 2^{n/2}/100$ pairs of vectors, and

$$\Pr_{A_b}[\exists q_1, q_2 \in Q \text{ such that } A_bq_1 = A_bq_2] \leq \sum_{q_1,q_2\in Q} \Pr_{A_b}[A_bq_1 = A_bq_2] \leq k^2 2^{-n/2} = 1/100.$$

Lemma 7. *Let f be a random draw from D_{YES}. Let $Q = \{q_1, q_2, \ldots, q_k\}$ be a set of $k = 2^{n/4}/10$ queries. Now, if the conditions in Lemmas 4 and 6 hold, then the vector $[f(q_1), f(q_2), \ldots, f(q_k)]$ is uniformly distributed.*

Proof. We choose a random matrix $A_b \in \mathbb{F}_2^{n/2\times n}$, and we extend to a full rank matrix A uniformly over all possible choices. Assuming the event from Lemma 6 holds, the vectors A_bq_i are all distinct. Since g is a uniformly random function, the values of $g(A_bq_i)$ are all independent and uniformly distributed, and it follows that the values of $f(q_i)$ are all independent and uniformly distributed as well.

Lemma 8. *Let f be a random draw from D_{NO}. Let Q be a set of $2^{n/4}/10$ queries. If the condition in Lemma 6 holds, the answers to the queries are uniformly distributed.*

Proof. Essentially the same as the latter portion of Lemma 7.

In order to prove a lower bound for adaptive testers, we can't assume that Q is a fixed query set, since for example q_2 depends on $f(q_1)$. A deterministic adaptive k-query algorithm is equivalent to a decision tree T of depth at most k. The internal nodes of T are labeled by query strings, and the leaves are labeled by "accept" and "reject". However, the best labeling of the leaves is easy to discuss. Given a decision tree T with unlabeled leaves, the best distinguisher one can get

by labeling the leaves is exactly $\|L_{\text{YES}} - L_{\text{NO}}\|_{TV}$; this is the result of labeling every leaf v with "accept" if $L_{\text{YES}}(v) > L_{\text{NO}}(v)$ and "reject" otherwise. As in [38], we define L_{YES} and L_{NO} to be distribution on leaves of T induced by a draw from D_{YES} and D_{NO}, respectively.

We fix a deterministic adaptive tester making at most k queries; equivalently, we fix a decision tree T of depth $k \leq 2^{n/4}/10$. Without loss of generality, we can assume that no string appears twice on any root-to-leaf path and the depth of every path is exactly k. It suffices to prove $\|L_{\text{YES}} - L_{\text{NO}}\|_{TV} \leq 1/3$.

Define L_{UNIF} to be the uniform distribution over the leaves of T. Consider a draw $f \sim D_{\text{YES}}$. Drawing A_b is the same as drawing a random $(n/2)$-dimensional subspace of $\mathbb{F}_2^n$. Consider the strings on nodes of a root-to-leaf path in T ending at the leaf v. By Lemma 7, all the strings on this path lie in different buckets, except with probability at most $1/100$ over the choice of A_b. Conditioned on this happening, the probability that f is consistent with the root-to-leaf path to v is exactly 2^{-k}, since g is a drawn uniformly at random. Thus, for each leaf v, we have $\Pr_{L_{\text{YES}}}[v \text{ is reached}] \geq (1 - 1/100)2^{-k}$. A similar argument shows that $\Pr_{L_{\text{NO}}}[v \text{ is reached}] \geq (1 - 1/100)2^{-k}$ as well.

The following lemma essentially appears in [38]:

Lemma 9. *Let D be a distribution over $\mathbb{F}_2^m$ that becomes the uniform distribution $\mathcal{U}$ conditioned on an event that happens with probability at least $99/100$. Then $\|D - \mathcal{U}\|_{TV} \leq 1/100$, where $\mathcal{U}$ is the uniform distribution over $\mathbb{F}_2^m$.*

Proof. Due to the conditioning, each element of $\mathbb{F}_2^m$ has probability mass at least $(99/100)2^{-m}$, so the elements with probability mass less than 2^{-m} contribute at most $1/2(1/100) = 1/200$ in total to the total variation distance. This lower bounding already takes up $99/100$ of the probability mass, so the elements with probability mass at least 2^{-m} contribute at most the remaining $1/2(1/100) = 1/200$ to the total variation distance. Thus $\|D - \mathcal{U}\|_{TV} \leq (1/2)(1/100 + 1/100) = 1/100$.

Theorem 8. *Any $(1/4, k)$-tester for testing membership in $\mathcal{L}(\mathcal{MM}_n)$ requires $2^{n/4}/10$ queries; that is, $k \geq 2^{n/4}/10$. This lower bounds holds for two-sided adaptive testers.*

Proof. The proof is via Yao's minimax principle. Let T be a decision tree of depth $2^{n/4}/10$ representing an adaptive deterministic tester, and let L_{YES} and L_{NO} be the distributions of leaves obtained by taking random draws from D_{YES} and D_{NO} respectively. The distributions L_{YES} and L_{NO} satisfy the conditions of Lemma 9 (by Lemmas 7 and 8), so $\|L_{\text{YES}} - L_{\text{UNIF}}\|_{TV} \leq 1/100$ and $\|L_{\text{NO}} - L_{\text{UNIF}}\|_{TV} \leq 1/100$. By the triangle inequality, $\|L_{\text{YES}} - L_{\text{NO}}\|_{TV} \leq 2/100 < 1/3$. It follows that the two distributions can not be distinguished using the adaptive tester characterized by decision tree T.

We can now prove Theorem 5:

Proof (Proof of Theorem 5). By Lemma 1, every function in $\mathcal{L}(\mathcal{MM}_n)$ is bent. By Proposition 1, every bent function is $(1/4)$-far from every $n/2$-dimensional function. Thus, we can use Yao's minimax principle and the same distributions

D_{YES} and D_{NO} as before to establish the theorem. The result now follows from mimicking the proof of Theorem 8.

Acknowledgments. We are grateful to Eric Blais, Amit Chakrabarti, Xiaoming Sun, and David Woodruff for helpful discussions. We would like to thank the anonymous referees for their comments and suggestions.

References

1. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. Journal of Computer and System Sciences 47, 549–595 (1993)
2. Rubinfeld, R., Sudan, M.: Robust characterizations of polynomials with applications to program testing. SIAM Journal on Computing 25, 252–271 (1996)
3. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. Journal of the ACM 45, 653–750 (1998)
4. Alon, N., Fischer, E., Newman, I., Shapira, A.: A combinatorial characterization of the testable graph properties: It's all about regularity. SIAM J. Comput. 39(1), 143–167 (2009)
5. Borgs, C., Chayes, J.T., Lovász, L., Sós, V., Szegedy, B., Vesztergombi, K.: Graph limits and parameter testing. In: STOC, pp. 261–270 (2006)
6. Bhattacharyya, A., Fischer, E., Hatami, H., Lovett, P.H.S.: Every locally characterized affine-invariant property is testable. In: STOC (to appear, 2013)
7. Kaufman, T., Sudan, M.: Algebraic property testing: The role of invariance. In: STOC, pp. 403–412 (2008)
8. Grigorescu, E., Kaufman, T., Sudan, M.: 2-transitivity is insufficient for local testability. Computational Complexity 22(1) (2013)
9. Grigorescu, E., Kaufman, T., Sudan, M.: Succinct representation of codes with applications to testing. SIAM J. Discrete Math. 26(4), 1618–1634 (2012)
10. Ben-Sasson, E., Sudan, M.: Limits on the rate of locally testable affine-invariant codes. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) APPROX/RANDOM 2011. LNCS, vol. 6845, pp. 412–423. Springer, Heidelberg (2011)
11. Ben-Sasson, E., Maatouk, G., Shpilka, A., Sudan, M.: Symmetric LDPC codes are not necessarily locally testable. In: CCC, pp. 55–65 (2011)
12. Ben-Sasson, E., Grigorescu, E., Maatouk, G., Shpilka, A., Sudan, M.: On sums of locally testable affine invariant properties. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) APPROX/RANDOM 2011. LNCS, vol. 6845, pp. 400–411. Springer, Heidelberg (2011)
13. Kaufman, T., Lovett, S.: New extension of the weil bound for character sums with applications to coding. In: FOCS, pp. 788–796 (2011)
14. Ben-Sasson, E., Ron-Zewi, N., Sudan, M.: Sparse affine-invariant linear codes are locally testable. In: FOCS, pp. 561–570 (2012)
15. Guo, A., Kopparty, S., Sudan, M.: New affine-invariant codes from lifting. In: ITCS, pp. 529–540 (2013)
16. Green, B.: A Szemerédi-type regularity lemma in abelian groups, with applications. Geom. Funct. Anal. 15(2), 340–376 (2005)
17. Král', D., Serra, O., Vena, L.: A combinatorial proof of the removal lemma for groups. Journal of Combinatorial Theory (A) 116(4), 971–978 (2009)
18. Shapira, A.: Green's conjecture and testing linear-invariant properties. In: Proc. 41st Annual ACM Symposium on the Theory of Computing, pp. 159–166 (2009)
19. Bhattacharyya, A., Grigorescu, E., Shapira, A.: A unified framework for testing linear-invariant properties. In: FOCS, pp. 478–487 (2010)

20. Blais, E., O'Donnell, R.: Lower bounds for testing function isomorphism. In: Proc. 25th Annual IEEE Conference on Computational Complexity, pp. 235–246 (2010)
21. Alon, N., Blais, E.: Testing boolean function isomorphism. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, pp. 394–405. Springer, Heidelberg (2010)
22. Chakraborty, S., García-Soriano, D., Matsliah, A.: Nearly tight bounds for testing function isomorphism. In: SODA, pp. 1683–1702 (2011)
23. Chakraborty, S., Fischer, E., García-Soriano, D., Matsliah, A.: Junto-symmetric functions, hypergraph isomorphism and crunching. In: CCC, pp. 148–158 (2012)
24. Blais, E., Weinstein, A., Yoshida, Y.: Partially symmetric functions are efficiently isomorphism-testable. In: FOCS, pp. 551–560 (2012)
25. Blais, E., Kane, D.: Tight bounds for testing k-linearity. In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) APPROX/RANDOM 2012. LNCS, vol. 7408, pp. 435–446. Springer, Heidelberg (2012)
26. McFarland, R.: A family of difference sets in non-cyclic groups. Journal of Combinatorial Theory (A) 15, 1–10 (1973)
27. Rothaus, O.: On "bent" functions. Journal of Combinatorial Theory (A) 20, 300–305 (1976)
28. Carlet, C., Guillot, P.: An alternate characterization of the bentness of binary functions, with uniqueness. Des. Codes Cryptography 14(2), 133–140 (1998)
29. Charnes, C., Dempwolff, U., Pieprzyk, J.: The eight variable homogeneous degree three bent functions. J. Discrete Algorithms 6(1), 66–72 (2008)
30. Dempwolff, U.: Geometric and design-theoretic aspects of semibent functions ii. Des. Codes Cryptography 62(2), 241–252 (2012)
31. Budaghyan, L., Carlet, C.: Ccz-equivalence of bent vectorial functions and related constructions. Des. Codes Cryptography 59(1-3), 69–87 (2011)
32. Carlet, C.: Relating three nonlinearity parameters of vectorial functions and building apn functions from bent functions. Des. Codes Cryptography 59(1-3), 89–109 (2011)
33. Neumann, T.: Bent functions. Master's thesis, University of Kaiserslautern (2006)
34. Bhattacharyya, A., Xie, N.: Lower bounds on testing triangle-freeness in Boolean functions. In: Proc. 21st ACM-SIAM Symposium on Discrete Algorithms, pp. 87–98 (2010)
35. Tokareva, N.: On the number of bent functions: lower bounds and hypotheses. IACR Cryptology ePrint Archive 2011, 83 (2011)
36. Parnas, M., Ron, D., Samorodnitsky, A.: Testing basic Boolean formulae. SIAM Journal on Discrete Mathematics 16(1), 20–46 (2003)
37. Fischer, E., Kindler, G., Ron, D., Safra, S., Samorodnitsky, A.: Testing juntas. Journal of Computer and System Sciences 68(4), 753–787 (2004)
38. Gopalan, P., O'Donnell, R., Servedio, R., Shpilka, A., Wimmer, K.: Testing Fourier dimensionality and sparsity. SIAM Journal on Computing 40(4), 1075–1100 (2011)
39. Wimmer, K., Yoshida, Y.: Testing linear-invariant function isomorphism. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 840–850. Springer, Heidelberg (2013)
40. Blais, E., Brody, J., Matulef, K.: Property testing lower bounds via communication complexity. Computational Complexity 21(2), 311–358 (2012)
41. Sun, X., Wang, C.: Randomized communication complexity for linear algebra problems over finite fields. In: Proc. 29th Annual Symposium on Theoretical Aspects of Computer Science, pp. 477–488 (2012)
42. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press (1997)

Randomness-Efficient Curve Samplers

Zeyu Guo*

Computer Science Department
California Institute of Technology
Pasadena, CA 91125
zguo@caltech.edu

Abstract. Curve samplers are sampling algorithms that proceed by viewing the domain as a vector space over a finite field, and randomly picking a low-degree curve in it as the sample. Curve samplers exhibit a nice property besides the sampling property: the restriction of low-degree polynomials over the domain to the sampled curve is still low-degree. This property is often used in combination with the sampling property and has found many applications, including PCP constructions, local decoding of codes, and algebraic PRG constructions.

The randomness complexity of curve samplers is a crucial parameter for its applications. It is known that (non-explicit) curve samplers using $O(\log N + \log(1/\delta))$ random bits exist, where N is the domain size and δ is the confidence error. The question of explicitly constructing randomness-efficient curve samplers was first raised in [TSU06] where they obtained curve samplers with near-optimal randomness complexity.

We present an explicit construction of low-degree curve samplers with *optimal* randomness complexity (up to a constant factor), sampling curves of degree $\left(m \log_q(1/\delta)\right)^{O(1)}$ in $\mathbb{F}_q^m$. Our construction is a delicate combination of several components, including extractor machinery, limited independence, iterated sampling, and list-recoverable codes.

1 Introduction

Randomness has numerous uses in computer science, and sampling is one of its most classical applications: Suppose we are interested in the size of a particular subset A lying in a large domain D. Instead of counting the size of A directly by enumeration, one can randomly draw a small sample from D and calculate the density of A in the sample. The approximated density is guaranteed to be close to the true density with probability $1 - \delta$ where δ is very small, known as the *confidence error*. This sampling technique is extremely useful both in practice and in theory.

One class of sampling algorithms, known as *curve samplers*, proceed by viewing the domain as a vector space over a finite field, and picking a random low-degree curve in it. Curve samplers exhibit the following nice property besides the sampling property: the restriction of low-degree polynomials over the domain to

* Supported by NSF CCF-1116111, NSF CCF-1038578 and BSF 2010120.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 575–590, 2013.

the sampled curve is still low-degree. This special property, combined with the sampling property, turns out to be useful in many settings, e.g local decoding of Reed-Muller codes and hardness amplification [STV01], PCP constructions [AS98, ALM+98, MR08], algebraic constructions of pseudorandom-generators [SU05, Uma03], extractor constructions [SU05, TSU06], and some pure complexity results (e.g. [SU06]).

The problem of explicitly constructing low-degree curve samplers was raised in [TSU06]. Typically, we are looking for low-degree curve samplers with small sample complexity (polylogrithmic in the domain size) and confidence error (polynomially small in the domain size), and we focus on minimizing the randomness complexity. The simplest way is picking a completely random low-degree curve whose sampling properties are guaranteed by tail bounds for limited independence. The randomness complexity of this method, however, is far from being optimal. The probabilistic method guarantees the existence of (non-explicit) low-degree curve samplers using $O(\log N + \log(1/\delta))$ random bits where N is the domain size and δ is the confidence error. The real difficulty, however, is to find an explicit construction matching this bound.

1.1 Previous Work

Randomness-efficient samplers (without the requirement that the sample points form a curve) are constructed in [CG89, Gil98, BR94, Zuc97]. In particular, [Zuc97] obtains explicit samplers with optimal randomness complexity (up to a $1+\gamma$ factor for arbitrary small $\gamma > 0$) using the connection between samplers and extractors. See [Gol11] for a survey of samplers.

Degree-1 curve samplers are also called *line samplers*. Explicit randomness-efficient line samplers are constructed in the PCP literature [BSSVW03, MR08], motivated by the goal of constructing almost linear sized PCPs. In [BSSVW03] line samplers are derandomized by picking a random point and a direction sampled from an ϵ-biased set, instead of two random points. An alternative way is suggested in [MR08] where directions are picked from a subfield. It is not clear, however, how to apply these techniques to higher degree curves.

In [TSU06] it was shown how to explicitly construct derandomized curve samplers with near-optimal parameters. Formally they obtained

- curve samplers picking curves of degree $(\log\log N + \log(1/\delta))^{O(\log\log N)}$ using randomness $O(\log N + \log(1/\delta)\log\log N)$, and
- curve samplers picking curves of degree $(\log(1/\delta))^{O(1)}$ using randomness $O(\log N + \log(1/\delta)(\log\log N)^{1+\gamma})$ for any constant $\gamma > 0$

for domain size N, field size $q \geq (\log N)^{\Theta(1)}$ and confidence error $\delta = N^{-\Theta(1)}$. Their work left the problem of explicitly constructing low-degree curve samplers (ideally picking curves of degree $O(\log_q(1/\delta))$) with essentially optimal $O(\log N + \log(1/\delta))$ random bits as a prominent open problem.

1.2 Main Results

We present an explicit construction of low-degree curve samplers with optimal randomness complexity (up to a constant factor). In particular, we show how to sample degree-$\left(m \log_q(1/\delta)\right)^{O(1)}$ curves in $\mathbb{F}_q^m$ using $O(\log N + \log(1/\delta))$ random bits for domain size $N = |\mathbb{F}_q^m|$ and confidence error $\delta = N^{-\Theta(1)}$. Before stating our main theorem, we first present the formal definition of samplers and curve samplers.

Samplers. Given a finite set $\mathcal{M}$ as the domain, the *density* of a subset $A \subseteq \mathcal{M}$ is $\mu(A) \stackrel{\text{def}}{=} \frac{|A|}{|\mathcal{M}|}$. For a collection of elements $\mathcal{T} = \{t_i : i \in I\} \in \mathcal{M}^I$ indexed by set I, the density of A in $\mathcal{T}$ is $\mu_{\mathcal{T}}(A) \stackrel{\text{def}}{=} \frac{|A \cap \mathcal{T}|}{|\mathcal{T}|} = \Pr_{i \in I}[t_i \in A]$.

Definition 1 (sampler). *A* sampler *is a function $S : \mathcal{N} \times \mathcal{D} \to \mathcal{M}$ where $|\mathcal{D}|$ is its* sample complexity *and $\mathcal{M}$ is its* domain*. We say S samples $A \subseteq \mathcal{M}$ with* accuracy error ϵ *and* confidence error δ *if $\Pr_{x \in \mathcal{N}}[|\mu_{S(x)}(A) - \mu(A)| > \epsilon] \leq \delta$ where $S(x) \stackrel{def}{=} \{S(x, y) : y \in \mathcal{D}\}$. We say S is an (ϵ, δ) sampler if it samples all subsets $A \subseteq \mathcal{M}$ with accuracy error ϵ and confidence error δ. The* randomness complexity *of S is $\log(|\mathcal{N}|)$.*

Definition 2 (curve/line sampler). *Let $\mathcal{M} = \mathbb{F}_q^D$ and $\mathcal{D} = \mathbb{F}_q$. The sampler $S : \mathcal{N} \times \mathcal{D} \to \mathcal{M}$ is a* degree-t curve sampler *if for all $x \in \mathcal{N}$, the function $S(x, \cdot) : \mathcal{D} \to \mathcal{M}$ is a curve (see Definition 3) of degree at most t over $\mathbb{F}_q$. When $t = 1$, S is also called a* line sampler*.*

Theorem 1 (main). *For any $\epsilon, \delta > 0$, integer $m \geq 1$, and sufficiently large prime power $q \geq \left(\frac{m \log(1/\delta)}{\epsilon}\right)^{\Theta(1)}$, there exists an explicit degree-t curve sampler for the domain $\mathbb{F}_q^m$ with $t = \left(m \log_q(1/\delta)\right)^{O(1)}$, accuracy error ϵ, confidence error δ, sample complexity q, and randomness complexity $O\left(m \log q + \log(1/\delta)\right) = O(\log N + \log(1/\delta))$ where $N = q^m$ is the domain size. Moreover, the curve sampler itself has degree $\left(m \log_q(1/\delta)\right)^{O(1)}$ as a polynomial map.*

Theorem 1 has better degree bound and randomness complexity compared with the constructions in [TSU06]. We remark that the degree bound, being $\left(m \log_q(1/\delta)\right)^{O(1)}$, is still sub-optimal compared with the lower bound $\log_q(1/\delta)$ (see Appendix A for the proof of this lower bound). However in many cases it is satisfying to achieve such a degree bound.

As an example, consider the following setting of parameters: domain size $N = q^m$, field size $q = (\log N)^{\Theta(1)}$, confidence error $\delta = N^{-\Theta(1)}$, and accuracy error $\epsilon = (\log N)^{-\Theta(1)}$. Note that this is the typical setting in PCP and other literature [ALM+98, AS98, STV01, SU05]. In this setting, we have the following corollary in which the randomness complexity is logarithmic and the degree is polylogarithmic.

Corollary 1. *Given domain size $N = |\mathbb{F}_q^m|$, accuracy error $\epsilon = (\log N)^{-\Theta(1)}$, confidence error $\delta = N^{-\Theta(1)}$, and large enough field size $q = (\log N)^{\Theta(1)}$, there exists an explicit degree-t curve sampler for the domain $\mathbb{F}_q^m$ with accuracy error ϵ, confidence error δ, randomness complexity $O(\log N)$, sample complexity q, and $t \leq (\log N)^c$ for some constant $c > 0$ independent of the field size q.*

It remains an open problem to explicitly construct curve samplers that have optimal randomness complexity $O(\log N + \log(1/\delta))$ (up to a constant factor), and sample curves with optimal degree bound $O(\log_q(1/\delta))$. It is also an interesting problem to achieve the optimal randomness complexity up to a $1+\gamma$ factor for any constant $\gamma > 0$ (rather than just an $O(1)$ factor), as achieved by [Zuc97] for general samplers. The standard techniques as in [Zuc97] are not directly applicable as they increase the dimension of samples and only yield $O(1)$-dimensional manifold samplers.

1.3 Techniques

Extractor machinery. It was shown in [Zuc97] that samplers are equivalent to *extractors*, objects that convert weakly random distributions into almost uniform distributions. Therefore the techniques of constructing extractors are extremely useful in constructing curve samplers. Our construction employs the technique of block source extraction [NZ96, Zuc97, SZ99]. In addition, we also use the techniques appeared in [GUV09], especially their constructions of *condensers*.

Limited independence. It is well known that points on a random degree-$(t-1)$ curve are t-wise independent. So we may simply pick a random curve and use tail inequalities to bound the confidence error. However, the sample complexity is too high, and hence we need to use the technique of *iterated sampling* to reduce the number of sample points.

Iterated sampling. Iterated sampling is a useful technique for explicitly constructing randomness-efficient samplers [BR94, TSU06]. The idea is first picking a large sample from the domain and then draw a sub-sample from the previous sample. The drawback of iterated sampling, however, is that it invests randomness twice while the confidence error does not shrink correspondingly. To remedy this problem, we add another ingredient into our construction, namely the technique of *error reduction*.

Error reduction via list-recoverable codes. We will use explicit list-recoverable codes (a strengthening of list-decodable codes [GI01]). More specifically, we will employ the list-recoverability from (folded) Reed-Solomon codes [GR08, GUV09]. List-recoverable codes provide a way of obtaining samplers with very small confidence error from those with mildly small confidence error. We refer to this transformation as *error reduction*, which plays a key role in our construction.

1.4 Sketch of the Construction

Our curve sampler is the composition of two samplers which we call the *outer sampler* and the *inner sampler* respectively. The outer sampler picks manifolds (see Definition 3) of dimension $O(\log m)$ from the domain $\mathcal{M} = \mathbb{F}_q^m$. The outer sampler has near-optimal randomness complexity but the sample complexity is large. To fix this problem, we employ the idea of iterated sampling. Namely we regard the manifold picked by the outer sampler as the new domain $\mathcal{M}'$, and then construct an inner sampler picking a curve from $\mathcal{M}'$ with small sample complexity.

The outer sampler is obtained by constructing an extractor and then using the extractor-sampler connection [Zuc97]. We follow the approach in [NZ96, Zuc97, SZ99]: Given an arbitrary random source with enough min-entropy, we will first use a *block source converter* to convert it into a *block source*, and then feed it to a *block source extractor*. In addition, we need to construct these components carefully so as to maintain the low-degree-ness. The way we construct the block source converter is different from those in [NZ96, Zuc97, SZ99] (as they are not in the form of low-degree polynomial maps), and is based on the Reed-Solomon condenser proposed in [GUV09]: To obtain one block, we simply feed the random source and a fresh new seed into the condenser, and let the output be the block. We show that this indeed gives a block source.

The inner sampler is constructed using techniques of iterated sampling and error reduction. We start with the basic curve samplers picking totally random curves, and then apply the error reduction as well as iterated sampling techniques repeatedly to obtain the desired inner sampler. Either of the two operations improves one parameter while worsening some other one: Iterated sampling reduces sample complexity but increases the randomness complexity, whereas error reduction reduces the confidence error but increases the sample complexity. Our construction applies the two techniques alternately such that (1) we keep the invariant that the confidence error is always exponentially small in the randomness complexity, and (2) the sample complexity is finally brought down to q.

Outline. The next section contains relevant definitions and some basic facts. Section 3 gives the construction of the outer sampler using block source extraction. Section 4 introduces the techniques of error reduction and iterated sampling, and then uses them to construct the inner sampler. These components are finally put together in Section 5 and yield the curve sampler construction.

2 Preliminaries

We denote the set of numbers $\{1, 2, \ldots, n\}$ by $[n]$. Given a prime power q, write $\mathbb{F}_q$ for the finite field of size q. Write $U_{n,q}$ for the uniform distribution over $\mathbb{F}_q^n$. Logarithms are taken with base 2 unless the base is explicitly specified.

Random variables and distributions are represented by upper-case letters whereas their specific values are represented by lower-case letters. Write $x \leftarrow X$

if x is sampled according to distribution X. The support of a distribution X over set S is $\mathrm{supp}(X) \stackrel{\text{def}}{=} \{x \in S : \Pr[X = x] > 0\}$. The statistical distance between distributions X, Y over set S is defined as $\Delta(X,Y) = \max_{T \subseteq S} |\Pr[X \in T] - \Pr[Y \in T]|$. We say X is ϵ-*close* to Y if $\Delta(X,Y) \leq \epsilon$.

For an event A, let $\mathbf{I}[A]$ be the indicator variable that evaluates to 1 if A occurs and 0 otherwise. For a random variable X and an event A that occurs with nonzero probability, define the *conditional distribution* $X|_A$ by $\Pr[X|_A = x] = \frac{\Pr[(X=x) \wedge A]}{\Pr[A]}$.

Manifolds and curves. Let $f : \mathbb{F}_q^d \to \mathbb{F}_q^D$ be a polynomial map. We may view f as D individual polynomials $f_i : \mathbb{F}_q^d \to \mathbb{F}_q$ describing its operation on each output coordinate, i.e., $f(x) = (f_1(x), \ldots, f_D(x))$ for all $x \in \mathbb{F}_q^d$. Such maps are called *curves* or *manifolds*, depending on the dimension d.

Definition 3 (manifold). *A manifold in $\mathbb{F}_q^D$ is a polynomial map $M : \mathbb{F}_q^d \to \mathbb{F}_q^D$ where $M_1, \ldots, M_D$ are d-variate polynomials over $\mathbb{F}_q$. We call d the* dimension *of M. An 1-dimensional manifold is also called a curve. A curve of degree 1 is also called a* line. *The degree of M is $\deg(M) \stackrel{\text{def}}{=} \max\{\deg(M_1), \ldots, \deg(M_D))\}$.*

We need the following lemma, generalizing the one in [TSU06]. The proof is the same as in [TSU06] and we omit it.

Lemma 1. *A manifold $f : \left(\mathbb{F}_{q^D}\right)^n \to \left(\mathbb{F}_{q^D}\right)^m$ of degree t, when viewed as a manifold $f : \left(\mathbb{F}_q^D\right)^n \to \left(\mathbb{F}_q^D\right)^m$, also has degree at most t.*

Basic line/curve samplers The simplest line (resp. curve) samplers are those picking completely random lines (resp. curves), as defined below. We call them *basic line* (resp. *curve*) *samplers*.

Definition 4 (basic line sampler). *Given $m \geq 1$ and prime power q, let $\mathsf{Line}_{m,q} : \mathbb{F}_q^{2m} \times \mathbb{F}_q \to \mathbb{F}_q^m$ be the line sampler that picks a completely random line in $\mathbb{F}_q^m$. Formally,*

$$\mathsf{Line}_{m,q}((a,b),y) \stackrel{def}{=} (a_1 y + b_1, \ldots, a_m y + b_m)$$

for $a = (a_1, \ldots, a_m), b = (b_1, \ldots, b_m) \in \mathbb{F}_q^m$ and $y \in \mathbb{F}_q$.

Definition 5 (basic curve sampler). *Given $m \geq 1$, $t \geq 4$ and prime power q, let $\mathsf{Curve}_{m,t,q} : \mathbb{F}_q^{tm} \times \mathbb{F}_q \to \mathbb{F}_q^m$ be the curve sampler that picks a completely random curve of degree $t-1$ in $\mathbb{F}_q^m$. Formally,*

$$\mathsf{Curve}_{m,t,q}((c_0, \ldots, c_{t-1}), y) \stackrel{def}{=} \left(\sum_{i=0}^{t-1} c_{i,1} y^i, \ldots, \sum_{i=0}^{t-1} c_{i,m} y^i \right)$$

for each $c_0 = (c_{0,1}, \ldots, c_{0,m}), \ldots, c_{t-1} = (c_{t-1,1}, \ldots, c_{t-1,m}) \in \mathbb{F}_q^m$ and $y \in \mathbb{F}_q$.

Remark 1. Note that $\mathsf{Line}_{m,q}$ has degree 2 and $\mathsf{Curve}_{m,t,q}$ has degree t as polynomial maps.

Lemma 2. *For $\epsilon > 0$, $m \geq 1$ and prime power q, $\mathsf{Line}_{m,q}$ is an $\left(\epsilon, \frac{1}{\epsilon^2 q}\right)$ line sampler.*

Lemma 3. *For $\epsilon > 0$, $m \geq 1$, $t \geq 4$ and sufficiently large prime power $q = (t/\epsilon)^{O(1)}$, $\mathsf{Curve}_{m,t,q}$ is an $\left(\epsilon, q^{-t/4}\right)$ sampler.*

Lemma 2 follows from Chebyshev's inequality and pairwise independence of points on a random line. Similarly, Lemma 3 follows from the tail inequalities for t-wise independence. See [TSU06, Lemma 2] for more details.

Extractors and condensers. A *(seeded) extractor* is an object that takes an imperfect random variable called the *(weakly) random source*, invests a small amount of randomness called the *seed*, and produces an output whose distribution is very close to the uniform distribution.

Definition 6 (q-ary min-entropy). *We say X has q-*ary min-entropy *k if for any $x \in S$, it holds that $\Pr[X = x] \leq q^{-k}$ (or equivalently, X has min-entropy $k \log q$).*

Definition 7 (condenser/extractor). *Given a function $f : \mathbb{F}_q^n \times \mathbb{F}_q^d \to \mathbb{F}_q^m$, we say f is an $k_1 \to_{\epsilon,q} k_2$* condenser *if for every distribution X with q-ary min-entropy k_1, $f(X, U_{d,q})$ is ϵ-close to a distribution with q-ary min-entropy k_2. We say f is a (k, ϵ, q)* extractor *if it is a $k \to_{\epsilon,q} m$ condenser.*

Remark 2. We are interested in extractors and samplers that are polynomial maps. For such an object f, we denote by $\deg(f)$ its degree as a polynomial map.

The following connection between extractors and samplers was observed in [Zuc97].

Theorem 2 ([Zuc97], restated). *Given a map $f : \mathbb{F}_q^n \times \mathbb{F}_q^d \to \mathbb{F}_q^m$, we have the following:*

1. *If f is a (k, ϵ, q) extractor, then it is also an (ϵ, δ) sampler where $\delta = 2q^{k-n}$.*
2. *If f is an $(\epsilon/2, \delta)$ sampler where $\delta = \epsilon q^{k-n}$, then it is also a (k, ϵ, q) extractor.*

3 Outer Sampler

In this section we construct a sampler whose randomness complexity is optimal up to a constant factor. We refer to it as the "outer sampler".

We need the machinery of block source extraction.

Definition 8 (block source [CG88]). *A random source $X = (X_1, \ldots, X_s)$ over $\mathbb{F}_q^{n_1} \times \cdots \times \mathbb{F}_q^{n_s}$ is a $(k_1, \ldots, k_s)$ q-*ary block source *if for any $i \in [s]$ and $(x_1, \ldots, x_{i-1}) \in \mathrm{supp}(X_1, \ldots, X_{i-1})$, the distribution $X_i|_{X_1 = x_1, \ldots, X_{i-1} = x_{i-1}}$ has q-ary min-entropy k_i. Each X_i is called a* block.

Definition 9 (block source extractor). *A function* $E : (\mathbb{F}_q^{n_1} \times \cdots \times \mathbb{F}_q^{n_s}) \times \mathbb{F}_q^d \to \mathbb{F}_q^m$ *is a* $((k_1, \ldots, k_s), \epsilon, q)$ block source extractor *if for any* $(k_1, \ldots, k_s)$ *q-ary block source* $(X_1, \ldots, X_s)$ *over* $\mathbb{F}_q^{n_1} \times \cdots \times \mathbb{F}_q^{n_s}$, $E((X_1, \ldots, X_s), U_{d,q})$ *is* ϵ*-close to* $U_{m,q}$.

The special structure of block sources allows us to compose several extractors and get a block source extractor, with a small amount of randomness invested.

Definition 10 (block source extraction via composition). *Let* $s \geq 1$ *be an integer and* $E_i : \mathbb{F}_q^{n_i} \times \mathbb{F}_q^{d_i} \to \mathbb{F}_q^{m_i}$ *be a map for each* $i \in [s]$. *Suppose that* $m_i \geq d_{i-1}$ *for all* $i \in [s]$, *where we set* $d_0 = 0$. *Define* $E = \mathsf{BlkExt}(E_1, \ldots, E_s)$ *as follows:*

$$E : (\mathbb{F}_q^{n_1} \times \cdots \times \mathbb{F}_q^{n_s}) \times \mathbb{F}_q^{d_s} \to (\mathbb{F}_q^{m_1 - d_0} \times \cdots \times \mathbb{F}_q^{m_s - d_{s-1}})$$
$$((x_1, \ldots, x_s), y_s) \mapsto (z_1, \ldots, z_s)$$

where for $i = s, \ldots, 1$, *we iteratively define* (y_{i-1}, z_i) *to be a partition of* $E_i(x_i, y_i)$ *into the prefix* $y_{i-1} \in \mathbb{F}_q^{d_{i-1}}$ *and the suffix* $z_i \in \mathbb{F}_q^{m_i - d_{i-1}}$.

The idea behind Definition 10 is to composes a chain of extractors E_i with decreasing output length and seed length. Then we use each E_i's output as the seed of the previous extractor E_{i-1}, so that the only seed the whole object actually needs is the (typically very short) one of E_s.

Lemma 4. *Let* $s \geq 1$ *be an integer and for each* $i \in [s]$, *let* $E_i : \mathbb{F}_q^{n_i} \times \mathbb{F}_q^{d_i} \to \mathbb{F}_q^{m_i}$ *be a* (k_i, ϵ_i, q) *extractor of degree* $t_i \geq 1$. *Then* $\mathsf{BlkExt}(E_1, \ldots, E_s)$ *is a* $((k_1, \ldots, k_s), \epsilon, q)$ *block source extractor of degree* t *where* $\epsilon = \sum_{i=1}^{s} \epsilon_i$ *and* $t = \prod_{i=1}^{s} t_i$.

The proof is standard and we defer it to the full version of this paper.

3.1 Block Source Conversion

Definition 11 (block source converter [NZ96]). *A function* $C : \mathbb{F}_q^n \times \mathbb{F}_q^d \to \mathbb{F}_q^{m_1} \times \cdots \times \mathbb{F}_q^{m_s}$ *is a* $(k, (k_1, \ldots, k_s), \epsilon, q)$ *block source converter if for any random source* X *over* $\mathbb{F}_q^n$ *with q-ary min-entropy* k, *the output* $C(X, U_{d,q})$ *is* ϵ*-close to a* $(k_1, \ldots, k_s)$ *q-ary block source.*

It was shown in [NZ96] that one can obtain a block by choosing a pseudorandom subset of bits of the random source. Yet the analysis is pretty delicate and cumbersome. Furthermore the resulting extractor does not have a nice algebraic structure. We observe that the following condenser from Reed-Solomon codes in [GUV09] can be used to obtain blocks and is a low-degree manifold.

Definition 12 (condenser from Reed-Solomon codes [GUV09]). *Let* $\zeta \in \mathbb{F}_q$ *be a generator of the multiplicative group* $\mathbb{F}_q^\times$. *Define* $\mathsf{RSCon}_{n,m,q} : \mathbb{F}_q^n \times \mathbb{F}_q \to \mathbb{F}_q^m$ *for* $n, m \geq 1$ *and prime power q:*

$$\mathsf{RSCon}_{n,m,q}(x, y) = \big(y, f_x(y), f_x(\zeta y), \ldots, f_x(\zeta^{m-2} y)\big)$$

where $f_x(Y) = \sum_{i=0}^{n-1} x_i Y^i$ *for* $x = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_q^n$.

Theorem 3 ([GUV09]). $\mathsf{RSCon}_{n,m,q}$ *is a* $m \to_{\epsilon,q} 0.99m$ *condenser for large enough* $q \geq (n/\epsilon)^{O(1)}$.

Remark 3. The condenser $\mathsf{RSCon}_{n,m,q}(x, y)$ is a degree-n manifold, since each monomial in any of its coordinate is of the form y or $x_i(\zeta^j y)^i$ for some $0 \leq i \leq n-1$.

We apply the above condenser to the source with independent seeds to obtain a block source.

Definition 13 (block source converter via condensing). *Given integers* $n, m_1, \ldots, m_s \geq 1$ *and prime power* q*, define the function* $\mathsf{BlkCnvt}_{n,(m_1,\ldots,m_s),q} : \mathbb{F}_q^n \times \mathbb{F}_q^s \to \mathbb{F}_q^{m_1+\cdots+m_s}$ *by*

$$\mathsf{BlkCnvt}_{n,(m_1,\ldots,m_s),q}(x, y) = (\mathsf{RSCon}_{n,m_1,q}(x, y_1), \ldots, \mathsf{RSCon}_{n,m_s,q}(x, y_s))$$

for $x \in \mathbb{F}_q^n$ *and* $y = (y_1, \ldots, y_s) \in \mathbb{F}_q^s$.

The function $\mathsf{BlkCnvt}_{n,(m_1,\ldots,m_s),q}$ is indeed a block source converter. The intuition is that conditioning on the values of the previous blocks, the random source X still has enough min-entropy, and hence we may apply the condenser to get the next block. Formally, we have the following statement whose proof is deferred to the full version of this paper.

Theorem 4. *For* $\epsilon > 0$*, integers* $s, n, m_1, \ldots, m_s \geq 1$ *and sufficiently large prime power* $q = (n/\epsilon)^{O(1)}$*,* $\mathsf{BlkCnvt}_{n,(m_1,\ldots,m_s),q}$ *is a* $(k, (k_1, \ldots, k_s), 3s\epsilon, q)$ *block source converter of degree* n *where* $k = \sum_{i=1}^{s} m_i + \log_q(1/\epsilon)$ *and each* $k_i = 0.99m_i$.

3.2 Construction of the Outer Sampler

By Lemma 2 and Theorem 2, the basic line samplers are also extractors.

Lemma 5. *For* $\epsilon > 0$*,* $m \geq 1$ *and prime power* q*,* $\mathsf{Line}_{m,q}$ *is a* (k, ϵ, q) *extractor of degree* 2 *where* $k = 2m - 1 + 3\log_q(1/\epsilon)$.

We employ Lemma 4 and compose the basic line samplers to get a block source extractor. It is then applied to a block source obtained from the block source converter.

Definition 14 (Outer Sampler). *For* $\delta > 0$*,* $m = 2^s$ *and prime power* q*, let* $n = 4m + \lceil \log_q(2/\delta) \rceil$*,* $d = s + 1$*, and* $d_i = 2^{s-i}$ *for* $i \in [s]$*. For* $i \in [s]$*, view* $\mathsf{Line}_{2,q^{d_i}} : \mathbb{F}_{q^{d_i}}^4 \times \mathbb{F}_{q^{d_i}} \to \mathbb{F}_{q^{d_i}}^2$ *as a manifold over* $\mathbb{F}_q$*:* $\mathsf{Line}_{2,q^{d_i}} : \mathbb{F}_q^{4d_i} \times \mathbb{F}_q^{d_i} \to \mathbb{F}_q^{2d_i}$*. Composing these line samplers* $\mathsf{Line}_{2,q^{d_i}}$ *for* $i \in [s]$ *gives the function* $\mathsf{BlkExt}(\mathsf{Line}_{2,q^{d_1}}, \ldots, \mathsf{Line}_{2,q^{d_s}}) : \mathbb{F}_q^{4d_1+\cdots+4d_s} \times \mathbb{F}_q \to \mathbb{F}_q^m$*. Finally, define function* $\mathsf{OuterSamp}_{m,\delta,q} : \mathbb{F}_q^n \times \mathbb{F}_q^d \to \mathbb{F}_q^m$ *such that for* $x \in \mathbb{F}_q^n$*,* $y \in \mathbb{F}_q^s$ *and* $y' \in \mathbb{F}_q$*,* $\mathsf{OuterSamp}_{m,\delta,q}(x, (y, y'))$ *equals*

$$\mathsf{BlkExt}(\mathsf{Line}_{2,q^{d_1}}, \ldots, \mathsf{Line}_{2,q^{d_s}})\big(\mathsf{BlkCnvt}_{n,(4d_1,\ldots,4d_s),q}(x, y), y'\big).$$

Theorem 5. *For any $\epsilon, \delta > 0$, integer $m \geq 1$, and sufficiently large prime power $q \geq (n/\epsilon)^{O(1)}$, $\mathsf{OuterSamp}_{m,\delta,q}$ is an (ϵ, δ) sampler of degree t where $d = O(\log m)$, $n = O\left(m + \log_q(1/\delta)\right)$ and $t = O\left(m^2 + m\log_q(1/\delta)\right)$.*

Proof. We first show that $\mathsf{OuterSamp}_{m,\delta,q}$ is a $(4m, \epsilon, q)$ extractor. Consider any random source X over $\mathbb{F}_q^n$ with q-ary min-entropy $4m$. Let s, d_i be as in Definition 14. Let $k_i = 4 \cdot 0.99 \cdot d_i$ for $i \in [s]$. Let $\epsilon_0 = \frac{\epsilon}{4s}$.

We have $(\sum_{i=1}^{s} 4d_i) + \log_q(1/\epsilon_0) \leq 4m$ for sufficiently large $q \geq (n/\epsilon)^{O(1)}$. So by Theorem 4, $\mathsf{BlkCnvt}_{n,(4d_1,\ldots,4d_s),q}$ is a $(4m, (k_1, \ldots, k_s), 3s\epsilon_0, q)$ block source converter. Therefore the distribution $\mathsf{BlkCnvt}_{n,(4d_1,\ldots,4d_s),q}(X, U_{s,q})$ is $3s\epsilon_0$-close to a $(k_1, \ldots, k_s)$ q-ary block source X'. Then $\mathsf{OuterSamp}_{m,\delta,q}(X, U_{d,q})$ is $3s\epsilon_0$-close to $\mathsf{BlkExt}(\mathsf{Line}_{2,q^{d_1}}, \ldots, \mathsf{Line}_{2,q^{d_s}})(X', U_{1,q})$.

By Lemma 5, $\mathsf{Line}_{2,q^{d_i}}$ is a $\left(k_i/d_i, \epsilon_0, q^{d_i}\right)$ extractor for $i \in [s]$ since $3 + 3\log_{q^{d_i}}(1/\epsilon_0) \leq 4 \cdot 0.99 = k_i/d_i$. Equivalently it is a (k_i, ϵ_0, q) extractor. By Lemma 4, $\mathsf{BlkExt}(\mathsf{Line}_{2,q^{d_1}}, \ldots, \mathsf{Line}_{2,q^{d_s}})$ is a $((k_1, \ldots, k_s), s\epsilon_0, q)$ block source extractor. Therefore $\mathsf{BlkExt}(\mathsf{Line}_{2,q^{d_1}}, \ldots, \mathsf{Line}_{2,q^{d_s}})(X', U_{1,q})$ is $s\epsilon_0$-close to $U_{m,q}$, which by the previous paragraph, implies that $\mathsf{OuterSamp}_{m,\delta,q}(X, U_{d,q})$ is $4s\epsilon_0$-close to $U_{m,q}$. By definition, $\mathsf{OuterSamp}_{m,\delta,q}$ is a $(4m, \epsilon, q)$ extractor. By Theorem 2, it is also an (ϵ, δ) sampler.

Finally, we have $d = s + 1 = O(\log m)$ and $n = O\left(m + \log_q(1/\delta)\right)$. By Lemma 1, each $\mathsf{Line}_{2,q^{d_i}}$ has degree 2 as a manifold over $\mathbb{F}_q$. Therefore by Lemma 4, the map $\mathsf{BlkExt}(\mathsf{Line}_{2,q^{d_1}}, \ldots, \mathsf{Line}_{2,q^{d_s}})$ has degree 2^s. By Theorem 4, $\mathsf{BlkCnvt}_{n,(4d_1,\ldots,4d_s),q}$ has degree n. Therefore $\mathsf{OuterSamp}_{m,\delta,q}$ has degree $n2^s = O\left(m^2 + m\log_q(1/\delta)\right)$. □

Remark 4. We assume m is a power of 2 above. For general m, simply pick $m' = 2^{\lceil \log m \rceil}$ and let $\mathsf{OuterSamp}_{m,\delta,q}$ be the composition of $\mathsf{OuterSamp}_{m',\delta,q}$ with the projection $\pi : \mathbb{F}_q^{m'} \to \mathbb{F}_q^m$ onto the first m coordinates. It yields an (ϵ, δ) sampler of degree t for $\mathbb{F}_q^m$ since π is linear, and approximating the density of a subset A in $\mathbb{F}_q^m$ is equivalent to approximating the density of $\pi^{-1}(A)$ in $\mathbb{F}_q^{m'}$.

4 Inner Sampler

The sampler $\mathsf{OuterSamp}_{m,\delta,q}$ has randomness complexity $O\left(m\log q + \log(1/\delta)\right)$ which is optimal up to a constant factor. Yet the sample complexity is large, being $q^{O(\log m)}$. We remedy this problem by composing it with an "inner sampler" with small sample complexity. Its construction is based on two techniques called error reduction and iterated sampling.

4.1 Error Reduction

Given $f : \mathbb{F}_q^n \times \mathbb{F}_q^d \to \mathbb{F}_q^m$, define $\mathrm{LIST}_f(T, \epsilon) \stackrel{\text{def}}{=} \left\{x \in \mathbb{F}_q^n : \Pr_y[f(x, y) \in T] > \epsilon\right\}$ for any $T \subseteq \mathbb{F}_q^m$ and $\epsilon > 0$. We are interesting in functions f exhibiting a "list-recoverability" property that the size of $\mathrm{LIST}_f(T, \epsilon)$ is kept small when T is not too large.

Definition 15. *A function* $f : \mathbb{F}_q^n \times \mathbb{F}_q \to \mathbb{F}_q^m$ *is* (ϵ, L, H) *list-recoverable if* $|\mathrm{LIST}_f(T, \epsilon)| \leq H$ *for all* $T \subseteq \mathbb{F}_q^m$ *of size at most* L.

We then define an operation $\star$ as follows.

Definition 16. *For functions* $f : \mathbb{F}_q^n \times \mathbb{F}_q^d \to \mathbb{F}_q^m$ *and* $S : \mathbb{F}_q^m \times \mathbb{F}_q^{d'} \to \mathbb{F}_q^{m'}$, *define* $S \star f : \mathbb{F}_q^n \times (\mathbb{F}_q^d \times \mathbb{F}_q^{d'}) \to \mathbb{F}_q^{m'}$ *such that* $(S \star f)(x, (y, y')) \stackrel{def}{=} S(f(x, y), y')$.

See Figure 1 for an illustration. The following lemma states that a sampler with mildly small confidence error, when composed with a list-recoverable function via the $\star$ operation, gives a sampler with very small confidence error.

Lemma 6. *Suppose* $f : \mathbb{F}_q^n \times \mathbb{F}_q^d \to \mathbb{F}_q^m$ *is* (ϵ_1, L, H) *list-recoverable, and* $S : \mathbb{F}_q^m \times \mathbb{F}_q^{d'} \to \mathbb{F}_q^{m'}$ *is an* $(\epsilon_2, L/q^m)$ *sampler. Then* $S \star f$ *is an* $(\epsilon_1 + \epsilon_2, H/q^n)$ *sampler.*

Proof. Let A be an arbitrary subset of $\mathbb{F}_q^{m'}$. Let $B = \{y \in \mathbb{F}_q^m : |\mu_{S(y)} - \mu(A)| > \epsilon_2\}$. By the sampling property of S, we have $|B| \leq (L/q^m) \cdot q^m = L$ and hence $|\mathrm{LIST}_f(B, \epsilon_1)| \leq H$. Therefore it suffices to show that for any $x \in \mathbb{F}_q^n \setminus \mathrm{LIST}_f(B, \epsilon_1)$, it holds that $|\mu_{(S \star f)(x)}(A) - \mu(A)| \leq \epsilon_1 + \epsilon_2$.

Fix $x \in \mathbb{F}_q^n \setminus \mathrm{LIST}_f(B, \epsilon_1)$. We have

$$\begin{aligned}\mu_{(S\star f)(x)}(A) = \Pr_{y,y'}[(S \star f)(x, (y, y')) \in A] &= \Pr_{y,y'}[S(f(x, y), y') \in A] \\ &= \mathbb{E}_y \left[\mu_{S(f(x,y))}(A)\right].\end{aligned}$$

Therefore

$$\begin{aligned}|\mu_{(S\star f)(x)}(A) - \mu(A)| = \left|\mathbb{E}_y \left[\mu_{S(f(x,y))}(A)\right] - \mu(A)\right| &\leq \mathbb{E}_y |\mu_{S(f(x,y))}(A) - \mu(A)| \\ &\leq \Pr_y[f(x, y) \in B] + \epsilon_2 \Pr_y[f(x, y) \notin B] \leq \epsilon_1 + \epsilon_2.\end{aligned}$$

To see the last two steps, note that $|\mu_{S(y)}(A) - \mu(A)| \leq \epsilon_2$ for $y \notin B$ by definition, and $\Pr_y[f(x, y) \in B] \leq \epsilon_1$ since $x \notin \mathrm{LIST}_f(B, \epsilon_1)$. □

It was shown in [GUV09] that the condenser $\mathsf{RSCon}_{n,m,q}$ (see Definition 12) enjoys the following list-recoverability property:

Theorem 6 ([GUV09]). *For sufficiently large* $q \geq (n/\epsilon)^{O(1)}$, *the function* $\mathsf{RSCon}_{n,m,q}$ *is* $\left(\epsilon, q^{0.99m}, q^m\right)$ *list-recoverable.*

Corollary 2. *For any* $n \geq m \geq 1$, $\epsilon, \epsilon' > 0$ *and sufficiently large prime power* $q = (n/\epsilon)^{O(1)}$, *suppose* $S : \mathbb{F}_q^m \times \mathbb{F}_q^d \to \mathbb{F}_q^{m'}$ *is an* $(\epsilon', q^{-0.01m})$ *sampler of degree* t, *then* $S \star \mathsf{RSCon}_{n,m,q}$ *is an* $(\epsilon + \epsilon', q^{m-n})$ *sampler of degree* nt.

Proof. Apply Lemma 6 and Theorem 6. Note that $\mathsf{RSCon}_{n,m,q}$ has degree n. Therefore $(S \star \mathsf{RSCon}_{n,m,q})(X, (Y, Y')) = S(\mathsf{RSCon}_{n,m,q}(X, Y), Y')$ has degree nt in its variables X, Y, Y'. □

4.2 Iterated Sampling

We introduce the operation $\circ$ denoting the composition of two samplers.

Definition 17. *(composed sampler). Given functions* $S_1 : \mathbb{F}_q^{n_1} \times \mathbb{F}_q^{d_1} \to \mathbb{F}_q^{d_0}$ *and* $S_2 : \mathbb{F}_q^{n_2} \times \mathbb{F}_q^{d_2} \to \mathbb{F}_q^{d_1}$*, define* $S_1 \circ S_2 : (\mathbb{F}_q^{n_1} \times \mathbb{F}_q^{n_2}) \times \mathbb{F}_q^{d_2} \to \mathbb{F}_q^{d_0}$ *such that* $(S_1 \circ S_2)((x_1, x_2), y) \stackrel{def}{=} S_1(x_1, S_2(x_2, y))$.

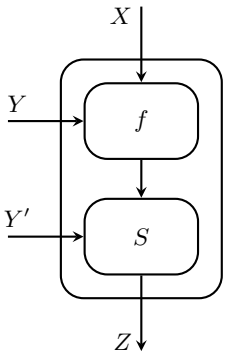


Fig. 1. The operation $S \star f$

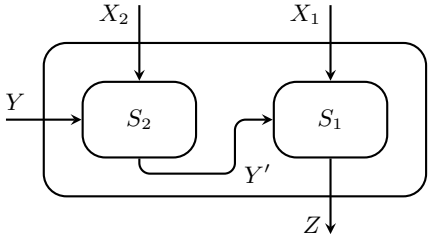


Fig. 2. The operation $S_1 \circ S_2$

See Figure 2 for an illustration. The composed sampler $S_1 \circ S_2$ first uses its randomness x_1 to get the sample $S_1(x_1) = \{S_1(x_1, y) : y \in \mathbb{F}_q\}$, and then uses its randomness x_2 to get the subsample $\{S_1(x_1, S_2(x_2, y)) : y \in \mathbb{F}_q\} \subseteq S_1(x_1)$. Intuitively, if S_1 and S_2 are good samplers then so is $S_1 \circ S_2$. This is indeed shown by [BR94, TSU06] and we formalize it as follows:

Lemma 7 ([BR94, TSU06]). *Let* $S_1 : \mathbb{F}_q^{n_1} \times \mathbb{F}_q^{d_1} \to \mathbb{F}_q^{d_0}$ *be an* (ϵ_1, δ_1) *sampler of degree* t_1 *and* $S_2 : \mathbb{F}_q^{n_2} \times \mathbb{F}_q^{d_2} \to \mathbb{F}_q^{d_1}$ *be an* (ϵ_2, δ_2) *sampler of degree* t_2*. Then* $S_1 \circ S_2 : (\mathbb{F}_q^{n_1} \times \mathbb{F}_q^{n_2}) \times \mathbb{F}_q^{d_2} \to \mathbb{F}_q^{d_0}$ *is an* $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ *sampler of degree* $t_1 t_2$*.*

Proof. Fix an arbitrary subset $A \subseteq \mathbb{F}_q^{d_0}$. Define $B(x) = \left\{z \in \mathbb{F}_q^{d_2} : S_1(x, z) \in A\right\}$ for $x \in \mathbb{F}_q^{n_1}$. Pick $x_1 \leftarrow U_{n_1,q}$ and $x_2 \leftarrow U_{n_2,q}$. If $\left|\mu_{S_1 \circ S_2((x_1,x_2))}(A) - \mu(A)\right| > \epsilon_1 + \epsilon_2$ occurs, then either $|\mu_{S_1(x_1)}(A) - \mu(A)| > \epsilon_1$, or $|\mu_{S_1 \circ S_2((x_1,x_2))}(A) - \mu_{S_1(x_1)}(A)| > \epsilon_2$ occurs. Call the two events E_1 and E_2 respectively.

Note that E_1 occurs with probability at most δ_1 by the sampling property of S_1. Also note that

$$\mu_{S_1 \circ S_2((x_1,x_2))}(A) = \Pr_y[S_1(x_1, S_2(x_2, y)) \in A] = \Pr_y[S_2(x_2, y) \in B(x_1)]$$
$$= \mu_{S_2(x_2)}(B(x_1))$$

whereas

$$\mu_{S_1(x_1)}(A) = \Pr_y[S_1(x_1, y) \in A] = \Pr_y[y \in B(x_1)] = \mu(B(x_1)).$$

So the probability that E_2 occurs is $\Pr_{x_1,x_2}[|\mu_{S_2(x_2)}(B(x_1)) - \mu(B(x_1))| > \epsilon_2]$ which is bounded by δ_2 by the sampling property of S_2. By the union bound, the event $\left|\mu_{S_1 \circ S_2((x_1,x_2))}(A) - \mu(A)\right| > \epsilon_1 + \epsilon_2$ occurs with probability at most $\delta_1 + \delta_2$, as desired.

Finally, we have $S_1 \circ S_2((X_1, X_2), Y)) = S_1(X_1, S_2(X_2, Y))$ which has degree $t_1 t_2$ in its variables X_1, X_2, Y since S_1 and S_2 have degree t_1 and t_2 respectively. □

4.3 Construction of the Inner Sampler

We use the basic curve samplers as the building blocks and apply the error reduction as well as iterated sampling repeatedly to obtain the inner sampler. The formal construction is as follows.

Definition 18 (inner sampler). *For $m \geq 1$, $\delta > 0$ and prime power q, pick $s = \lceil \log m \rceil$ and let $d_i = 2^{s-i}$ for $0 \leq i \leq s$. Let $n_i = 16^i$ for $0 \leq i \leq s-1$ and $n_s = 16^s + 20\left\lceil \log_q(1/\delta) \right\rceil$. Define $S_i : \mathbb{F}_q^{n_i d_i} \times \mathbb{F}_q^{d_i} \to \mathbb{F}_q^m$ for $0 \leq i \leq s$ as follows:*

- $S_0 : \mathbb{F}_q \times \mathbb{F}_q^{d_0} \to \mathbb{F}_q^m$ *projects (x, y) onto the first m coordinates of y.*
- $S_i \stackrel{def}{=} \left(S_{i-1} \star \mathsf{RSCon}_{\frac{n_i}{4}, 2n_{i-1}, q^{d_i}}\right) \circ \mathsf{Curve}_{3, \frac{n_i}{4}, q^{d_i}}$ *for $i = 1, \ldots, s$.*

Finally, let $\mathsf{InnerSamp}_{m,\delta,q} \stackrel{def}{=} S_s$.

Theorem 7. *For any $\epsilon, \delta > 0$, integer $m \geq 1$ and large enough prime power $q \geq \left(\frac{m \log(1/\delta)}{\epsilon}\right)^{O(1)}$, $\mathsf{InnerSamp}_{m,\delta,q} : \mathbb{F}_q^n \times \mathbb{F}_q \to \mathbb{F}_q^m$ is an (ϵ, δ) sampler of degree t where $n = O\left(m^{O(1)} + \log_q(1/\delta)\right)$ and $t = O\left(m^{O(\log m)} \log_q^2(1/\delta)\right)$.*

Proof. Let $\epsilon' = \frac{\epsilon}{2s}$ and d_i, n_i, S_i be as in Definition 18 for $0 \leq i \leq s$. We will prove that each S_i is an (ϵ_i, δ_i) sampler of degree t_i where $\epsilon_i = 2i\epsilon'$, $\delta_i = q^{-n_i d_i/20}$, and $t_i = \prod_{j=1}^{i} \left(\frac{n_j}{4}\right)^2$. The theorem follows by noting that $\mathsf{InnerSamp}_{m,\delta,q} = S_s$, $\epsilon = \epsilon_s$, $\delta \geq \delta_s$, and $t = t_s$.

Induct on i. The case $i = 0$ is trivial. Consider the case $i > 0$ and assume the claim holds for all $i' < i$. By the induction hypothesis, S_{i-1} is an $(\epsilon_{i-1}, \delta_{i-1})$ sampler of degree t_{i-1}.

Note that $\delta_{i-1} = q^{-n_{i-1}d_{i-1}/20} \leq q^{-0.01 n_{i-1} d_{i-1}}$. By Corollary 2, $S_{i-1} \star \mathsf{RSCon}_{\frac{n_i}{4}, 2n_{i-1}, q^{d_i}}$ is an $\left(\epsilon_{i-1} + \epsilon', q^{n_{i-1}d_{i-1} - (n_i/4)d_i}\right)$ sampler of degree $\frac{n_i}{4} \cdot t_{i-1}$. By Lemma 3, $\mathsf{Curve}_{3, \frac{n_i}{4}, q^{d_i}}$ is an $\left(\epsilon', q^{-n_i d_i/16}\right)$ sampler of degree $\frac{n_i}{4}$. Finally by Lemma 7, the function

$$S_i = \left(S_{i-1} \star \mathsf{RSCon}_{\frac{n_i}{4}, 2n_{i-1}, q^{d_i}}\right) \circ \mathsf{Curve}_{3, \frac{n_i}{4}, q^{d_i}}$$

is an $\left(\epsilon_{i-1} + 2\epsilon', q^{n_{i-1}d_{i-1} - (n_i/4)d_i} + q^{-n_i d_i/16}\right)$ sampler of degree $\left(\frac{n_i}{4}\right)^2 \cdot t_{i-1}$. It remains to check that

$$\epsilon_i = \epsilon_{i-1} + 2\epsilon', \quad \delta_i \geq q^{n_{i-1}d_{i-1} - (n_i/4)d_i} + q^{-n_i d_i/16} \quad \text{and} \quad t_i = \left(\frac{n_i}{4}\right)^2 \cdot t_{i-1}.$$

which hold by the choices of parameters. □

5 Putting It Together

We compose the outer sampler and the inner sampler to get the desired curve sampler.

Definition 19. *For $m \geq 1$, $\delta > 0$ and prime power q, define*

$$\mathsf{Samp}_{m,\delta,q} \stackrel{def}{=} \mathsf{OuterSamp}_{m,\delta/2,q} \circ \mathsf{InnerSamp}_{d,\delta/2,q}.$$

Theorem 8 (Theorem 1 restated). *For any $\epsilon, \delta > 0$, integer $m \geq 1$ and sufficiently large prime power $q \geq \left(\frac{m \log(1/\delta)}{\epsilon}\right)^{O(1)}$, the function $\mathsf{Samp}_{m,\delta,q} : \mathbb{F}_q^n \times \mathbb{F}_q \to \mathbb{F}_q^m$ is an (ϵ, δ) sampler of degree t where $n = O\left(m + \log_q(1/\delta)\right)$ and $t = \left(m \log_q(1/\delta)\right)^{O(1)}$. In particular, $\mathsf{Samp}_{m,\delta,q}$ is an (ϵ, δ) degree-t curve sampler.*

Proof. By Theorem 5, $\mathsf{OuterSamp}_{m,\delta/2,q} : \mathbb{F}_q^{n_1} \times \mathbb{F}_q^d \to \mathbb{F}_q^m$ is an $(\epsilon/2, \delta/2)$ sampler of degree t_1 where $d = O(\log m)$, $n_1 = O\left(m + \log_q(1/\delta)\right)$ and $t_1 = O\left(m^2 + m \log_q(1/\delta)\right)$.

By Theorem 7, $\mathsf{InnerSamp}_{d,\delta/2,q} : \mathbb{F}_q^{n_2} \times \mathbb{F}_q \to \mathbb{F}_q^d$ is an $(\epsilon/2, \delta/2)$ sampler of degree t_2 with parameters $n_2 = O\left((\log m)^{O(1)} + \log_q(1/\delta)\right)$ and $t_2 = O\left((\log m)^{O(\log \log m)} \log_q^2(1/\delta)\right)$.

Finally, Lemma 7 implies that $\mathsf{Samp}_{m,\delta,q}$ is an (ϵ, δ) sampler of degree t with $n = n_1 + n_2 = O\left(m + \log_q(1/\delta)\right)$ and $t = t_1 t_2 = \left(m \log_q(1/\delta)\right)^{O(1)}$. A fortiori, it is a degree-t curve sampler since the degree of $\mathsf{Samp}_{m,\delta,q}(x, \cdot)$ is bounded by the degree of $\mathsf{Samp}_{m,\delta,q}$ for all $x \in \mathbb{F}_q^n$. □

Acknowledgements. The author is grateful to Chris Umans for his support and many helpful discussions.

References

[ALM+98] Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM 45(3), 501–555 (1998)

[AS98] Arora, S., Safra, S.: Probabilistic checking of proofs: a new characterization of NP. J. ACM 45(1), 70–122 (1998)

[BR94] Bellare, M., Rompel, J.: Randomness-efficient oblivious sampling. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, SFCS 1994, pp. 276–287. IEEE Computer Society, Washington, DC (1994)

[BSSVW03] Ben-Sasson, E., Sudan, M., Vadhan, S., Wigderson, A.: Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing, STOC 2003, pp. 612–621. ACM, New York (2003)

[CG88] Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. SIAM J. Comput. 17, 230–261 (1988)

[CG89] Chor, B., Goldreich, O.: On the power of two-point based sampling. J. Complex. 5(1), 96–106 (1989)

[Din07] Dinur, I.: The PCP theorem by gap amplification. J. ACM 54(3) (June 2007)

[GI01] Guruswami, V., Indyk, P.: Expander-based constructions of efficiently decodable codes. In: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, FOCS 2001, pp. 658–667. IEEE Computer Society, Washington, DC (2001)

[Gil98] Gillman, D.: A Chernoff bound for random walks on expander graphs. SIAM J. Comput. 27(4), 1203–1220 (1998)

[Gol11] Goldreich, O.: A sample of samplers: A computational perspective on sampling. In: Goldreich, O. (ed.) Studies in Complexity and Cryptography. LNCS, vol. 6650, pp. 302–332. Springer, Heidelberg (2011)

[GR08] Guruswami, V., Rudra, A.: Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. IEEE Trans. Inf. Theor. 54(1), 135–150 (2008)

[GUV09] Guruswami, V., Umans, C., Vadhan, S.: Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. J. ACM 56, 20:1–20:34 (2009)

[MR08] Moshkovitz, D., Raz, R.: Sub-constant error low degree test of almost-linear size. SIAM J. Comput. 38(1), 140–180 (2008)

[NZ96] Nisan, N., Zuckerman, D.: Randomness is linear in space. J. Comput. Syst. Sci. 52, 43–52 (1996)

[Rei08] Reingold, O.: Undirected connectivity in log-space. J. ACM 55(4), 17:1–17:24 (2008)

[RTS00] Radhakrishnan, J., Ta-Shma, A.: Bounds for dispersers, extractors, and depth-two superconcentrators. SIAM J. Discret. Math. 13(1), 2–24 (2000)

[STV01] Sudan, M., Trevisan, L., Vadhan, S.: Pseudorandom generators without the XOR lemma. J. Comput. Syst. Sci. 62(2), 236–266 (2001)

[SU05] Shaltiel, R., Umans, C.: Simple extractors for all min-entropies and a new pseudorandom generator. J. ACM 52(2), 172–216 (2005)

[SU06] Shaltiel, R., Umans, C.: Pseudorandomness for approximate counting and sampling. Comput. Complex. 15(4), 298–341 (2006)

[SZ99] Srinivasan, A., Zuckerman, D.: Computing with very weak random sources. SIAM J. Comput. 28, 1433–1459 (1999)

[TSU06] Ta-Shma, A., Umans, C.: Better lossless condensers through derandomized curve samplers. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 177–186. IEEE Computer Society, Washington, DC (2006)

[Uma03] Umans, C.: Pseudo-random generators for all hardnesses. J. Comput. Syst. Sci. 67(2), 419–440 (2003)

[Zuc97] Zuckerman, D.: Randomness-optimal oblivious sampling. In: Proceedings of the Workshop on Randomized Algorithms and Computation, pp. 345–367. John Wiley & Sons, Inc., New York (1997)

A Lower Bounds on the Degree of Sampled Curves

We present the following lower bound on the degree of curves sampled by a curve sampler:

Theorem 9. *Let $S : \mathcal{N} \times \mathbb{F}_q \to \mathbb{F}_q^m$ be an (ϵ, δ) degree-t curve sampler where $m \geq 2$, $\epsilon < 1/2$ and $\delta < 1$. Then $t = \Omega\left(\log_q(1/\delta) + 1\right)$.*

Proof. Clearly $t \geq 1$. Suppose $S = (S_1, \ldots, S_m)$ and define $S' = (S_1, S_2)$. Let $\mathcal{C}$ be the set of curves of degree at most t in $\mathbb{F}_q^2$. Then $|\mathcal{C}| = q^{2(t+1)}$. Consider the map $\tau : \mathcal{N} \to \mathcal{C}$ that sends x to $S'(x, \cdot)$. We can pick $k = \lfloor q/2 \rfloor$ curves $C_1, \ldots, C_k \in \mathcal{C}$ such that the union of their preimages

$$B \stackrel{\text{def}}{=} \bigcup_{i=1}^{k} \tau^{-1}(C_i) = \bigcup_{i=1}^{k} \{x : S'(x, \cdot) = C_i\}$$

has size at least $\frac{k|\mathcal{N}|}{|\mathcal{C}|} = \frac{k|\mathcal{N}|}{q^{2(t+1)}}$.

Define $A \subseteq \mathbb{F}_q^m$ by

$$A \stackrel{\text{def}}{=} \{C_i(y) : i \in [k],\ y \in \mathbb{F}_q\} \times \mathbb{F}_q^{m-2},$$

i.e., let A be the set of points in $\mathbb{F}_q^m$ whose first two coordinates are on at least one curve C_i. We have $|A| \leq kq^{m-1}$ and hence $\mu(A) \leq k/q \leq 1/2 < 1 - \epsilon$. On the other hand, it follows from the definition of A that we have $S(x, y) \in A$ for all $x \in B$ and $y \in \mathbb{F}_q$. So $\mu_{S(x)}(A) = 1$ for all $x \in B$. Then

$$\delta \geq \Pr\left[|\mu_{S(x)}(A) - \mu(A)| > \epsilon\right] \geq \frac{|B|}{|\mathcal{N}|} \geq \frac{k}{q^{2(t+1)}}$$

and hence $t \geq \max\left\{1, \frac{1}{2}\log_q(k/\delta) - 1\right\} = \Omega\left(\log_q(1/\delta) + 1\right)$. □

We remark that the condition $m \geq 2$ is necessary in Theorem 9 since when $m = 1$, the sampler S with $S(x, y) = y$ for all $x \in \mathcal{N}$ and $y \in \mathbb{F}_q$ is a $(0, 0)$ degree-1 curve sampler.

Combinatorial Limitations of Average-Radius List Decoding*

Venkatesan Guruswami** and Srivatsan Narayanan

Computer Science Department
Carnegie Mellon University
{venkatg,srivatsa}@cs.cmu.edu

Abstract. We study certain combinatorial aspects of list-decoding, motivated by the exponential gap between the known upper bound (of $O(1/\gamma)$) and lower bound (of $\Omega_p(\log(1/\gamma))$) for the list-size needed to list decode up to error fraction p with rate γ away from capacity, i.e., $1-h(p)-\gamma$ (here $p \in (0, \frac{1}{2})$ and $\gamma > 0$). Our main result is the following:

> We prove that in any binary code $C \subseteq \{0,1\}^n$ of rate $1-h(p)-\gamma$, there must exist a set $\mathcal{L} \subset C$ of $\Omega_p(1/\sqrt{\gamma})$ codewords such that the average distance of the points in $\mathcal{L}$ from their centroid is at most pn. In other words, there must exist $\Omega_p(1/\sqrt{\gamma})$ codewords with low "average radius." The standard notion of list-decoding corresponds to working with the *maximum* distance of a collection of codewords from a center instead of *average* distance. The average-radius form is in itself quite natural; for instance, the classical Johnson bound in fact implies average-radius list-decodability.

The remaining results concern the standard notion of list-decoding, and help clarify the current state of affairs regarding combinatorial bounds for list-decoding:

- We give a short simple proof, over all fixed alphabets, of the above-mentioned $\Omega_p(\log(1/\gamma))$ lower bound. Earlier, this bound followed from a complicated, more general result of Blinovsky.
- We show that one *cannot* improve the $\Omega_p(\log(1/\gamma))$ lower bound via techniques based on identifying the zero-rate regime for list-decoding of constant-weight codes. On a positive note, our $\Omega_p(1/\sqrt{\gamma})$ lower bound for average-radius list-decoding circumvents this barrier.
- We exhibit a "reverse connection" between the existence of constant-weight and general codes for list-decoding, showing that the best possible list-size, as a function of the gap γ of the rate to the capacity limit, is the same up to constant factors for both constant-weight codes (whose weight is bounded away from p) and general codes.
- We give simple second moment based proofs that w.h.p. a list-size of $\Omega_p(1/\gamma)$ is needed for list-decoding *random* codes from errors as well as erasures. For *random linear* codes, the corresponding list-size bounds are $\Omega_p(1/\gamma)$ for errors and $\exp(\Omega_p(1/\gamma))$ for erasures.

* Full version can be found at http://eccc.hpi-web.de/report/2012/017/

** Research supported in part by NSF grant CCF 0953155 and a Packard Fellowship.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 591–606, 2013.

1 Introduction

The list-decoding problem for an error-correcting code $C \subseteq \Sigma^n$ consists of finding the set of all codewords of C with Hamming distance at most pn from an input string $y \in \Sigma^n$. Though it was originally introduced in early work of Elias and Wozencraft [6,15] in the context of estimating the decoding error probability for random error models, recently the main interest in list-decoding has been for adversarial error models. List decoding enables correcting up to a factor two more worst-case errors compared to algorithms that are always restricted to output a unique answer, and this potential has even been realized algorithmically [10,8].

In this work, we are interested in some fundamental combinatorial questions concerning list-decoding, which highlight the important tradeoffs in this model. Fix $p \in (0, \frac{1}{2})$ and a positive integer L. We say that a binary code $C \subseteq \{0,1\}^n$ is (p, L) list-decodable if every Hamming ball of radius pn has *less than* L codewords. Here, p corresponds to the error-fraction and L to the list-size needed by the error-correction algorithm. Note that (p, L) list-decodability imposes a sparsity requirement on the distribution of codewords in the Hamming space. A natural combinatorial question that arises in this context is to place bounds on the largest size of a code meeting this requirement. In particular, an outstanding open question is to characterize the maximum rate (defined to be the limiting ratio $\frac{1}{n} \log |C|$ as $n \to \infty$) of a (p, L) list-decodable code.

By a simple volume packing argument, it can be shown that a (p, L) list-decodable code has rate at most $1 - h(p) + o(1)$. (Throughout, for $z \in [0, \frac{1}{2}]$, we use $h(z)$ to denote the binary entropy function at z.) Indeed, picking a random center x, the Hamming ball $\mathbf{B}(x, pn)$ contains at least $|C| \cdot \binom{n}{pn} 2^{-n}$ codewords in expectation. Bounding this by $(L-1)$, we get the claim. On the positive side, in the limit of large L, the rate of a (p, L) list-decodable code approaches the optimal $1 - h(p)$. More precisely, for any $\gamma > 0$, there exists a $(p, 1/\gamma)$ list-decodable code of rate at least $1 - h(p) - \gamma$. In fact, a random code of rate $1 - h(p) - \gamma$ is $(p, 1/\gamma)$ list-decodable w.h.p. [16,7], and a similar result holds for random linear codes (with list-size $O_p(1/\gamma)$) [9]. In other words, a dense random packing of $2^{(1-h(p)-\gamma)n}$ Hamming balls of radius pn (and therefore volume $\approx 2^{h(p)n}$ each) is "near-perfect" w.h.p. in the sense that no point is covered by more than $O_p(1/\gamma)$ balls.

The determination of the best asymptotic code rate of binary (p, L) list-decodable codes as p, L are held fixed and the block length grows is wide open for every choice of $p \in (0, \frac{1}{2})$ and integer $L \geqslant 1$. However, we *do* know that for each fixed $p \in (0, \frac{1}{2})$, this rate approaches $1 - h(p)$ in the limit as $L \to \infty$. To understand this rate of convergence as a function of list-size L, following [9], let us define $L_{p,\gamma}$ to be the minimum integer L such that there exist (p, L) list-decodable codes of rate $1 - h(p) - \gamma$ for infinitely many block lengths n (the quantity γ is the "gap" to "list-decoding capacity"). In [1], Blinovsky showed that a (p, L) list-decodable code has rate at most $1 - h(p) - 2^{-\Theta_p(L)}$. In particular, this implies that for any finite L, a (p, L) list-decodable code has rate strictly below the optimal $1 - h(p)$. Stated in terms of $L_{p,\gamma}$, his result implies the corollary $L_{p,\gamma} \geqslant \Omega_p(\log(1/\gamma))$ for rates γ-close to capacity. We provide a short and simple

proof of this corollary in Section 4. Our proof works almost as easily over non-binary alphabets. (Blinovsky's subsequent proof for the non-binary case in [3,4] involved substantial technical effort. However, his results also give non-trivial bounds for every finite L, as opposed to just the growth rate of $L_{p,\gamma}$.)

Observe the exponential gap (in terms of the dependence on γ) between the $O(1/\gamma)$ upper bound and $\Omega_p(\log(1/\gamma))$ lower bounds on the quantity $L_{p,\gamma}$. Despite being a basic and fundamental question about sphere packings in the Hamming space and its direct relevance to list-decoding, there has been no progress on narrowing this asymptotic gap in the 25 years since the works of Zyablov-Pinsker [16] and Blinovsky [1]. This is the motivating challenge driving this work.

1.1 Prior Work on List-Size Lower Bounds

We now discuss some lower bounds (besides Blinovsky's general lower bound) on list-size that have been obtained in restricted cases.

Rudra shows that the $O_p(1/\gamma)$ bound obtained via the probabilistic method for random codes is, in fact, tight up to constant factors [14]. Formally, there exists $L = \Omega_p(1/\gamma)$ such that a random code of rate $1 - h(p) - \gamma$ is *not* (p, L) list-decodable w.h.p. His proof uses near-capacity-achieving codes for the binary symmetric channel, the existence of which is promised by Shannon's theorem, followed by a second moment argument. We give a simpler proof of this result via a more direct use of the second moment method. This has the advantage that it works uniformly for random general as well as random linear codes, and for channels that introduce errors as well as erasures.

Guruswami and Vadhan [12] consider the problem of establishing list-size bounds when the channel may corrupt close to half the bits, that is, when $p = \frac{1}{2} - \varepsilon$, and more generally $p = 1 - 1/q - \varepsilon$ for codes over an alphabet of size q. (Note that decoding is impossible if the channel could corrupt up to a half fraction of bits.) They show that there exists $c > 0$ such that for all $\varepsilon > 0$ and all block lengths n, any $(\frac{1}{2} - \varepsilon, c/\varepsilon^2)$ list-decodable code contains $O_\varepsilon(1)$ codewords. For p bounded away from $\frac{1}{2}$ (or $1-1/q$ in the q-ary case), their methods do not yield any nontrivial list-size lower bound as a function of gap γ to list-decoding capacity.

1.2 Our Main Results

We have already mentioned our new proof of the $\Omega(\log(1/\gamma))$ list-size lower bound for list-decoding general codes, and the asymptotically optimal list-size lower bound for random (and random linear) codes.

Our main result concerns an average-radius variant of list-decoding. This variant was implicitly used in [1,12] en route their list-size lower bounds for standard list-decoding. In this work, we formally abstract this notion of *average-radius list-decodability*: a code is (p, L) *average-radius list-decodable* if for every L codewords, the *average* distance of their centroid from the L codewords exceeds pn. Note that this is a stronger requirement than (p, L) list-decodability where only the *maximum* distance from any center point to the L codewords must exceed pn.

We are able to prove nearly tight bounds on the achievable rate of a (p, L) average-radius list-decodable code. To state our result formally, denote by $L^{\mathsf{avg}}_{p,\gamma}$ the minimum L such that there exists a (p, L) average-radius list-decodable code family of rate $1-h(p)-\gamma$. A simple random coding argument shows that a random code of $1-h(p)-\gamma$ is $(p, 1/\gamma)$ average-radius list-decodable (matching the list-decodability of random codes). That is, $L^{\mathsf{avg}}_{p,\gamma} \leqslant 1/\gamma$. Our main technical result is a lower bound on the list-size that is polynomially related to the upper bound, namely $L^{\mathsf{avg}}_{p,\gamma} \geqslant \Omega_p(\gamma^{-1/2})$.

We remark that the classical Johnson bound in coding theory in fact proves the average-radius list-decodability of codes with good minimum distance — namely, a binary code of relative distance δ is $(J(\delta - \delta/L), L)$ average-radius list-decodable, where $J(z) = (1 - \sqrt{1-2z})/2$ for $z \in [0, \frac{1}{2}]$. (This follows from a direct inspection of the proof of the Johnson bound [11].) Also, one can show that if a binary code is $(\frac{1}{2} - 2^i\varepsilon, O(1/(2^{2i}\varepsilon^2))$ list-decodable for all $i = 0, 1, 2, \ldots$, then it is also $(\frac{1}{2} - 2\varepsilon, O(1/\varepsilon^2))$ average-radius list-decodable [5]. This shows that at least in the high noise regime, there is some reduction between these notions. Further, a suitable soft version of average-radius list-decodability can be used to construct matrices with a certain restricted isometry property [5]. For these reasons, we feel that average-radius list-decodability is a natural notion to study, even beyond treating it as a vehicle to understand (standard) list-decoding.

1.3 Our other Results

We also prove several other results that clarify the landscape of combinatorial limitations of list-decodable codes. Many results showing rate limitations in coding theory proceed via a typical approach in which they pass to a constant weight $\lambda \in (p, \frac{1}{2}]$; i.e., they restrict the codewords to be of weight exactly λn. They show that under this restriction, a code with the stated properties must have a constant number of codewords (that is, asymptotically *zero rate*). Mapping this bound back to the unrestricted setting one gets a rate upper bound of $1-h(\lambda)+o(1)$ for the original problem. For instance, the Elias-Bassalygo bound for rate R vs. relative distance δ is of this nature (here λ is picked to be the Johnson radius for list-decoding for codes of relative distance δ).

The above is also the approach taken in Blinovsky's work [1] as well as that of [12]. We show that such an approach does not and *cannot* give any bound better than Blinovsky's $\Omega_p(\log(1/\gamma))$ bound for $L_{p,\gamma}$. More precisely, for any $\lambda \geqslant p + 2^{-b_p L}$ for some $b_p > 0$, we show that there exists a (p, L) (average-radius) list-decodable code of rate $\Omega_{p,L}(1)$. Thus in order to improve the lower bound, we *must* be able to handle codes of strictly positive rate, and cannot deduce the bound by pinning down the zero-rate regime of constant-weight codes. This perhaps points to why improvements to Blinovsky's bounds have been difficult. On a positive note, we remark that we *are* able to effect such a proof for average-radius list-decoding (some details follow next).

To describe the method underlying our list-size lower bound for average-radius list-decoding, it is convenient to express the statement as an upper bound on rate in terms of list-size L. Note that a list-size lower bound of $L \geqslant \Omega_p(1/\sqrt{\gamma})$

for (p, L) average-radius list-decodable codes of rate $1 - h(p) - \gamma$ amounts to proving an upper bound of $1 - h(p) - \Omega_p(1/L^2)$ on the rate of (p, L) average-radius list-decodable codes. Our proof of such an upper bound proceeds by first showing a rate upper bound of $h(\lambda) - h(p) - a_p/L^2$ for such codes when the codewords are all restricted to have weight λn, for a suitable choice of λ, namely $\lambda = p + a'_p/L$. To map this back to the original setting (with no weight restrictions on codewords), one simply notes that every (p, L) average-radius list-decodable code of rate R contains as a subcode, a translate of a constant λn-weight subcode of rate $R - (1 - h(\lambda))$. (The second step uses a well-known argument.)

Generally speaking, by passing to a constant-weight subcode, one can translate combinatorial results on limitations of constant-weight codes to results showing limitations for the case of general codes. But this leaves open the possibility that the problem of showing limitations of constant-weight codes may be harder than the corresponding problem for general codes, or worse still, have a different answer making it impossible to solve the problem for general codes via the methodology of passing to constant-weight codes. We show that for the problem of list-decoding this is fortunately not the case, and there is, in fact, a "reverse connection" of the following form: A rate upper bound of $1 - h(p) - \gamma$ for (p, L) list-decodable codes implies a rate upper bound of $h(\lambda) - h(p) - \left(\frac{\lambda - p}{\frac{1}{2} - p}\right)\gamma$ for (p, L) list-decodable codes whose codewords must all have Hamming weight λn. A similar claim holds also for average-radius list-decodability, though we don't state it formally.

1.4 Our Proof Techniques

Our proofs in this paper employ variants of the standard probabilistic method. We show an extremely simple probabilistic argument that yields a $\Omega_p(\log(1/\gamma))$ bound on the list-size of a standard list-decodable code; we emphasize that this is qualitatively the tightest known bound in this regime.

For the "average-radius list-decoding" problem that we introduce, we are able to improve this list-size bound to $\Omega_p(1/\sqrt{\gamma})$. The proof is based on the idea that instead of picking the "bad list-decoding center" x uniformly at random, one can try to pick it randomly very close to a special codeword c^*, and this still gives similar guarantees on the number of near-by codewords. Now since the quantity of interest is the average radius, including this close-by codeword in the list gives enough savings for us. In order to estimate the probability that a typical codeword c belongs to the list around x, we write this probability explicitly as a function of the Hamming distance between c^* and c, which is then lower bounded using properties of hypergeometric distributions and Taylor approximations for the binary entropy function.

For limitations of list-decoding random codes, we define a random variable W that counts the number of "violations" of the list-decoding property of the code. We then show that W has a exponentially large mean, around which it is concentrated w.h.p. This yields that the code cannot be list-decodable with high probability, for suitable values of rate and list-size parameters. We skip the formal statement of these results and their proofs in this version (due to space restrictions); these can be found in the full version.

1.5 Organization

We define some useful notation and the formal notion of average-radius list-decodability in Section 2. Our main list-size lower bound for average-radius list-decoding appears in Section 3. We give our short proof of Blinovsky's lower bounds for binary and general alphabets in Section 4. Our results about the zero-error rate regime for constant-weight codes and the connection between list-decoding bounds for general codes and constant-weight codes appear in Section 5. For reasons of space, many of the proofs, and all results on list-size lower bounds for random codes, are skipped and can be found in the full version.

2 Notation and Preliminaries

2.1 List Decoding

We recall some standard terminology regarding error-correcting codes. Let $[n]$ denote the index set $\{1, 2, \ldots, n\}$. For $q \geqslant 2$, let $[q]$ denote the set $\{0, 1, \ldots, q-1\}$. A *q-ary code* refers to any subset $C \subseteq [q]^n$, where n is the *blocklength* of C. We will mainly focus on the special case of binary codes corresponding to $q = 2$. The rate $R = R(C)$ is defined to be $\frac{\log |C|}{n \log q}$. For $x \in [q]^n$ and $S \subseteq [n]$, the restriction of x to the coordinates in S is denoted $x|_S$. Let $\mathrm{Supp}(x) := \{i \in [n] \ : \ x_i \neq 0\}$. A *subcode* of C is a subset C' of C. We say that C is a *constant-weight code* with weight $w \in [0, n]$, if all its codewords have weight exactly w. (Such codes are studied in Section 5.)

For $x, y \in [q]^n$, define the *Hamming distance* between x and y, denoted $d(x, y)$, to be the number of coordinates in which x and y differ. The *(Hamming) weight* of x, denoted $\mathrm{wt}(x)$, is $d(\mathbf{0}, x)$, where $\mathbf{0}$ is the vector in $[q]^n$ with zeroes in all coordinates. The *(Hamming) ball* of radius r centered at x, denoted $\mathbf{B}(x, r)$, is the set $\{y \in [q]^n \ : \ d(x, y) \leqslant r\}$. In this paper, we need the following nonstandard measure of distance of a (small) "list" $\mathcal{L}$ of vectors from a "center" x: For a nonempty $\mathcal{L} \subseteq [q]^n$, define

$$D_{\mathsf{max}}(x, \mathcal{L}) := \max\{d(x, y) \ : \ y \in \mathcal{L}\},$$

and

$$D_{\mathsf{avg}}(x, \mathcal{L}) := \mathbf{E}_{y \in \mathcal{L}} \left[d(x, y) \right] = \frac{1}{|\mathcal{L}|} \sum_{y \in \mathcal{L}} d(x, y).$$

We formalize the error recovery capability of the code using list-decoding.

Definition 1. *Fix $0 < p < \frac{1}{2}$ and a positive integer L. Let C be a q-ary code with blocklength n.*

1. *C is said to be* (p, L) list-decodable *if for all $x \in [q]^n$, $\mathbf{B}(x, pn)$ contains at most $L - 1$ codewords from C. Equivalently, for any x and any list $\mathcal{L} \subseteq C$ of size at least L, we have $D_{\mathsf{max}}(x, \mathcal{L}) > pn$.*
2. *C is said to be* (p, L) average-radius list-decodable *if for any x and $\mathcal{L}$ as in Item 1, we have $D_{\mathsf{avg}}(x, \mathcal{L}) > pn$.*

For constant-weight codes, it is convenient to augment the notation with the weight parameter:

Definition 2. *Let p, L, q, n, C be as in Definition 1, and let $0 < \lambda \leqslant \frac{1}{2}$. C is said to be $(\lambda; p, L)$* (average-radius) list-decodable *if C is (p, L) (average-radius) list-decodable, and every codeword in C has weight exactly λn.*

We remark that the list-decodability property is standard in literature. Moreover, while the notion of (p, L) *average-radius* list-decodability is formally introduced by this paper, it is already implicit in [1,2,12].

Since the max-distance of a list from a center always dominates its average distance, every (p, L) average-radius list-decodable code is also (p, L) list-decodable. That is, average-radius list-decodability is a syntactically stronger property than its standard counterpart, and hence any limitation we establish for the standard list-decodable codes also carries over for average-radius list-decodability.

Following (and extending) the notation in [9], we make the following definitions to quantify the tradeoffs in the different parameters of a code: the rate R, the error-correction radius p, list-size L, and the weight λ of the codewords (for constant-weight codes). Further, for general codes (without the constant-weight restriction), it is usually more convenient to replace the rate R by the parameter $\gamma := 1 - h(p) - R$; this measures the "gap" to the "limiting rate" or the "capacity" of $1 - h(p)$ for $(p, O(1))$ list-decodable codes.

Fix $p, \lambda \in (0, \frac{1}{2}]$ such that $p < \lambda$, $R \in (0, 1)$, and a positive integer L.

Definition 3. *1. Say that the triple $(p, L; R)$ is* achievable for list-decodable codes *if there exist (p, L) list-decodable codes of rate R for infinitely many lengths n.*
Define $R_{p,L}$ to be the supremum over R such that $(p, L; R)$ is achievable for list-decodable codes, and $\gamma_{p,L} := 1 - h(p) - R_{p,L}$. Also define $L_{p,\gamma}$ to be the least integer L such that $(p, L; 1 - h(p) - \gamma)$ is achievable.

*2. (***For constant weight codes.***) Say that the 4-tuple $(\lambda; p, L; R)$ is achievable if there exists $(\lambda; p, L)$ list-decodable codes of rate R. Define $R_{p,L}(\lambda)$ to be the supremum rate R for which the 4-tuple $(\lambda; p, L; R)$ is achievable.*

We can also define analogous quantities for average-radius list-decoding (denoted by a superscript avg), but to prevent notational clutter, we will not explicitly do so. Throughout this paper, p is treated as a fixed constant in $(0, \frac{1}{2})$, and we will not attempt to optimize the dependence of our bounds on p.

2.2 Standard Distributions and Functions

In this paper, we use 'log' for logarithms to base 2 and 'ln' for natural logarithms. Also, to avoid cumbersome notation, we often denote b^z by $\exp_b(z)$. Standard asymptotic notation (big O, little o, and big Omega) is employed liberally in this paper; when subscripted by a parameter (typically p), the notation hides a constant depending arbitrarily on the parameter.

Our proofs also make repeated use of *hypergeometric distributions*, which we review here for the sake of completeness, as well as to set the notation. Suppose

a set contains n objects, exactly $m < n$ of which are *m*arked, and suppose we *s*ample $s < n$ objects uniformly at random from the set *without replacement*. Then the random variable T counting the number of marked objects in the sample follows the hypergeometric distribution with parameters (n, m, s). A simple counting argument shows that, for $t \leqslant \min\{m, s\}$,

$$\mathbf{Pr}[T = t] = \frac{\binom{m}{t}\binom{n-m}{s-t}}{\binom{n}{s}}.$$

We will denote the above expression by $f(n, m, s, t)$. By convention, $f(n, m, s, t)$ is set to 0 if $n < \max\{m, s\}$ or $t > \min\{m, s\}$. Hypergeometric distributions satisfy a useful symmetry property:

Lemma 1. *For all integers n, m, s with $n \geqslant \max\{m, s\}$, the hypergeometric distribution with parameters (n, m, s) is identical to that with parameters (n, s, m). That is, for all t, we have $f(n, m, s, t) = f(n, s, m, t)$.*

Throughout this paper, we are especially concerned with the asymptotic behaviour of binomial coefficients, which is characterized in terms of the *binary entropy function*, defined as $h(z) := -z \log z - (1 - z) \log(1 - z)$. We will use the following standard estimate without proof: For $z \in (0, 1)$ and $n \to \infty$, if zn is an integer, then

$$\exp_2(h(z)n - o(n)) \leqslant \binom{n}{zn} \leqslant \sum_{i=0}^{zn} \binom{n}{i} \leqslant \exp_2(h(z)n).$$

3 Bounds for Average-Radius List-Decodability

In this section, we bound the rate of a (p, L) average-radius list-decodable code as:

$$1 - h(p) - \frac{1}{L} - o(1) \leqslant R \leqslant 1 - h(p) - \frac{a_p}{L^2} + o(1),$$

where a_p is a constant depending only on p. (Here p is a fixed constant bounded away from 0 and $\frac{1}{2}$.) Note that, ignoring the dependence on p, the corresponding upper and lower bounds on $\gamma := 1 - h(p) - R$ are polynomially related.

We first state the rate lower bound.

Theorem 1. *Fix $p \in (0, \frac{1}{2})$ and a positive integer L. Then, for all $\varepsilon > 0$ and all sufficiently large lengths n, there exists a (p, L) average-radius list-decodable code of rate at least $1 - h(p) - 1/L - \varepsilon$.*

In fact, a random code of the above rate has the desired property w.h.p. This calculation is routine and omitted here, and can be found in the full version.

We now show an upper bound of $1 - h(p) - a_p/L^2$ on the rate of a (p, L) average-radius list-decodable code. As stated in the Introduction, the main idea behind the construction is that instead of picking the "bad list decoding center" x uniformly at random, we pick it randomly *very close to a designated codeword*

c^* (which itself is a uniformly random element from C). Now as long as we are guaranteed to find a list of $L-1$ other codewords near x, we can include c^* in our list to lower the average radius of the list.

However formalizing the above intuition into a proof is nontrivial, since our restriction of the center x to be very close to c^* introduces statistical dependencies while analyzing the number of codewords near x. We are able to control these dependencies, but this requires some heavy calculations involving the entropy function and hypergeometric distribution.

We are now ready to state our main result establishing a rate upper bound for (p, L) average-radius list-decodable codes. In fact, the bulk of the work is to show an analogous upper bound for the special case of a constant-weight code C, i.e., all codewords have weight exactly λn, for some $\lambda \in (p, \frac{1}{2}]$. We can then map this bound for general codes using a standard argument (given in Lemma 2).

Theorem 2 (Main theorem). *Fix $p \in (0, \frac{1}{2})$, and let L be a sufficiently large positive integer. Then there exist $a_p, a'_p > 0$ (depending only on p) such that the following holds (for sufficiently large lengths n):*

1. *If C is a (p, L) average-radius list-decodable code, then C has rate at most $1 - h(p) - a_p/L^2 + o(1)$.*
2. *For $\lambda := p + a'_p/L$, if C is a $(\lambda; p, L)$ average-radius list-decodable code, then C has rate at most $h(\lambda) - h(p) - a_p/L^2 + o(1)$.*

As already mentioned in Section 1.3, the second claim readily implies the first via the following well-known argument (a partial converse to this statement for list-decoding will be given in Section 5):

Lemma 2. *Let $\lambda \in (p, \frac{1}{2}]$ be such that λn is an integer. If C is a (p, L) average-radius list-decodable code of rate $R = 1 - h(p) - \gamma$, then there exists a $(\lambda; p, L)$ average-radius list-decodable code C' of rate at least $R' - o(1)$, where $R' := h(\lambda) - h(p) - \gamma$.*

Proof: For a random center x, let $C'(x)$ be the subcode of C containing the codewords c with $d(x, c) = \lambda n$. The expected size of $C'(x)$ at least $|C| \cdot \binom{n}{\lambda n} 2^{-n}$, which, for the assumed value of R, is $\exp_2(R'n - o(n))$; thus for some x, $C'(x)$ has rate at least $R' - o(1)$. The claim follows by translating $C'(x)$ by $-x$. □

Before we proceed to the proof of (the first part of) Theorem 2, we will establish the following folklore result, whose proof illustrates our idea in a simple case.

Lemma 3 (A warm-up lemma). *Fix p, λ so that $p < \lambda \leqslant \frac{1}{2}$. Then, if C is a $(\lambda; p, L)$ list-decodable code, then C has rate at most $h(\lambda) - h(p) + o(1)$.*

Proof: The main idea behind the proof is that a random center of a *particular weight* (carefully chosen) is close to a large number of codewords in expectation. Pick a random subset $S \subseteq [n]$ of coordinates of size αn, with $\alpha := (\lambda - p)/(1 - 2p)$, and let $\overline{S} = [n] \setminus S$. (The motivation for this choice of α will be clear shortly.) Define the center x be the indicator vector of S, so that $\mathrm{Supp}(x) = S$.

Consider the set $\mathcal{L}$ of codewords $c \in C$ such that $\mathrm{wt}(c|_S) \geqslant (1-p)\alpha n$; this is our candidate bad list of codewords. Then each $c \in \mathcal{L}$ is close to c:

$$d(x, c) = (\alpha n - \mathrm{wt}(c|_S)) + \mathrm{wt}(c|_{\overline{S}}) \leqslant \alpha p n + (\lambda - \alpha(1-p))n = (\lambda - \alpha(1-2p))n,$$

which equals pn for the given choice of α. Hence the size of $\mathcal{L}$ is a lower bound on the list-size of the code.

We complete the proof by computing $\mathbf{E}\,|\mathcal{L}|$. For any fixed $c \in C$, the random variable $\mathrm{wt}(c|_S)$ follows the hypergeometric distribution with parameters $(n, \lambda n, \alpha n)$, , which is identical to the hypergeometric distribution with parameters $(n, \alpha n, \lambda n)$ (see Lemma 1). Hence the probability that c is included in the list $\mathcal{L}$ is at least

$$f(n, \alpha n, \lambda n, \alpha(1-p)n) := \frac{\binom{\alpha n}{(1-p)\alpha n}\binom{(1-\alpha)n}{(\lambda-\alpha(1-p))n}}{\binom{n}{\lambda n}} = \frac{\binom{\alpha n}{p\alpha n}\binom{(1-\alpha)n}{p(1-\alpha)n}}{\binom{n}{\lambda n}},$$

where the second step holds because of the identity $\lambda - (1-p)\alpha = p(1-\alpha)$, which holds for our particular choice of α. As $n \to \infty$, this is equal to

$$\exp_2\left(\alpha n h(p) + (1-\alpha)n h(p) - h(\lambda)n - o(n)\right) = \exp_2((h(p) - h(\lambda) - o(1))n).$$

Thus, by linearity of expectations, the expected size of $\mathcal{L}$ is at least $|C| \cdot \exp_2((h(p) - h(\lambda) - o(1))n)$. On the other hand, the (p, L) list-decodability of C says that $|\mathcal{L}| \leqslant L$ (with probability 1). Comparing these lower and upper bounds on $\mathbf{E}\,|\mathcal{L}|$ yields the claim. □

Proof of Theorem 2 (part 2): At a high level, we proceed as in the proof of Lemma 3, but in addition to the bad list $\mathcal{L}$ of codewords, we will a special codeword $c^* \in C$ such that $d(x, c^*)$ is *much smaller than the codewords in* $\mathcal{L}$. Then defining $\mathcal{L}^*$ to consist of c^* and $(L-1)$ other codewords from $\mathcal{L}$, we see that the *average* distance of $\mathcal{L}^*$ is much smaller than before, thus enabling us to obtain an improved rate bound.

We now provide the details. Pick a uniformly random codeword $c^* \in C$. Let $S \subseteq [n]$ be a random subset of $\mathrm{Supp}(c^*)$ of size βn, where the parameter β is chosen appropriately later[1] (this plays the role of α in Lemma 3). Also, let x be the indicator vector of S.

As before, consider the set $\mathcal{L}$ of codewords $c \in C$ such that $\mathrm{wt}(c|_S) \geqslant (1-p)|S|$. For a fixed $c \in C$, the random variable $\mathrm{wt}(c|_S)$ follows the hypergeometric distribution with parameters $(\lambda n, (\lambda - \delta)n, \beta n)$, where $\delta = \delta(c^*, c)$ is defined by $d(c^*, c) := 2\delta n$. (Observe that the normalization ensures that $0 \leqslant \delta \leqslant \lambda$ for all pairs $c^*, c \in C$.) To see this, notice that we are sampling βn coordinates from $\mathrm{Supp}(c^*)$ without replacement, and that $\mathrm{wt}(c|_S)$ simply counts the number of coordinates picked from $\mathrm{Supp}(c^*) \cap \mathrm{Supp}(c)$ (the size of this intersection is exactly $(\lambda - \delta)n$). Thus, conditioned on c^*, the probability that a fixed $c \in C$ is included in $\mathcal{L}$ is

$$Q(\delta) := \sum_{w=(1-p)\beta n}^{\beta n} f(\lambda n, (\lambda-\delta)n, \beta n, w). \tag{1}$$

By linearity of expectations, and taking expectations over c^*, the expected size of $\mathcal{L}$ can be written as $\mathbf{E}_{c^* \in C}\left[\sum_{c \in C} Q(\delta(c^*, c))\right] = |C| \cdot \mathbf{E}\; Q(\delta(c^*, c))$, where

[1] The reader might find it helpful to think of β as $O(1/L)$; roughly speaking, this translates to a rate upper bound of $h(\lambda) - h(p) - \Omega(\beta/L)$.

both c^* and c are picked uniformly at random from C. The bulk of the work lies in obtaining a lower bound on this expectation, which we state below.

Claim. For $A_1 := (1-p)\log\left(\frac{1-p}{\lambda}\right) + p\log\left(\frac{p}{1-\lambda}\right)$ and $A_2 = \frac{5}{p^2}$, we have

$$\mathbf{E}\, Q(\delta(c^*, c)) \geqslant \exp_2\left(-(A_1\beta + A_2\beta^2 + o(1))n\right).$$

Proof Sketch: A standard application of the Cauchy-Schwarz inequality shows that $\mathbf{E}\,\delta \leqslant \lambda(1-\lambda)$, and hence Markov's inequality implies that

$$\delta \leqslant \lambda(1-\lambda) + \frac{1}{n} = \lambda(1-\lambda) + o(1)$$

with probability at least $1/n$. Moreover, since $Q(\delta)$ is a monotone decreasing function of δ, we have

$$\mathbf{E}\, Q(\delta) \geqslant \frac{1}{n} \cdot Q(\lambda(1-\lambda) + o(1)).$$

The rest of the proof is technical and involves lower bounding the right hand side using properties of binomial coefficients and Taylor approximations for the binary entropy function. Due to lack of space, we skip the detailed calculations, which can be found in the full version. □

Therefore, as before, if the code C has rate $A_1\beta + A_2\beta^2 + o(1)$ (for a suitable $o(1)$ term), the list $\mathcal{L}$ has size at least L in expectation. Fix some choice of c^* and S such that $|\mathcal{L}| \geqslant L$. Let $\mathcal{L}^*$ be any list containing c^* and $L-1$ other codewords from $\mathcal{L}$; we are interested in $D_{\mathsf{avg}}(x, \mathcal{L}^*)$. Clearly, $d(x, c^*) = (\lambda - \beta)n$. On the other hand, for $c \in \mathcal{L}^* \setminus \{c^*\}$, we can bound its distance from x as: $d(x,c) \leqslant \beta pn + (\lambda - \beta(1-p))n = (\lambda - \beta(1-2p))n$, where the two terms are respectively the contribution by S and $[n] \setminus S$. Averaging these L distances, we get that

$$D_{\mathsf{avg}}(x, \mathcal{L}^*) \leqslant (\lambda - \beta(1 - 2p + 2p/L))\, n.$$

Now, we pick β so that this expression is at most pn; i.e., set

$$\beta := \frac{\lambda - p}{1 - 2p + 2p/L}. \tag{2}$$

(Compare with the choice of α in Lemma 3.) For this choice of β, the list $\mathcal{L}^*$ violates the average-radius list-decodability property of C.

Thus the rate of a (p, L) average-radius list-decodable code is upper bounded by $R \leqslant A_1\beta + A_2\beta^2 + o(1)$, where β is given by (2). Further technical manipulations brings this to the following more convenient form: If $L > \frac{2p}{1-2p}$, then

$$R \leqslant (h(\lambda) - h(p)) - \frac{B_1(\lambda - p)}{L} + B_2(\lambda - p)^2 + o(1).$$

for some constants B_1 and B_2 depending only on p. (See the full version for a detailed calculation.) Setting $\lambda := p + B_1/(2B_2L)$, the rate is upper bounded by $R \leqslant h(\lambda) - h(p) - B_1^2/(4B_2L^2) + o(1)$. □

4 Bounds for (Standard) List-Decodability

In this section, we consider the rate vs. list-size tradeoff for the traditional list-decodability notion. For the special case when the fraction of errors is close to $\frac{1}{2}$, [12] showed that any code family of growing size correcting up to $\frac{1}{2} - \varepsilon$ fraction of errors must have a list-size $\Omega(1/\varepsilon^2)$, which is optimal up to constant factors. When p is bounded away from 1/2, Blinovsky [1,3] gives the best known bounds on the rate of a (p, L) list-decodable code. His results imply (see [14] for the calculations) that any (p, L) list-decodable code of rate $1 - h(p) - \gamma$ has list-size L at least $\Omega_p(\log(1/\gamma))$. We give a short and simple proof of this latter claim in this section.

Theorem 3 ([1,3]).

1. *Suppose C is $(\lambda; p, L)$ list-decodable code with $\lambda = p + \frac{1}{2}p^L$. Then $|C| \leqslant 2L^2/p$, independent of its blocklength n. (In particular, the rate approaches 0 as $n \to \infty$.)*
2. *Any (p, L) list-decodable code has rate at most $1 - h(p) - \Omega_p(p^L)$.*

Proof: For the first part, assume for the sake of contradiction that $|C| > 2L^2/p$. Pick a random L-tuple of codewords (without replacement) $\mathcal{L} = \{c_1, c_2, \ldots, c_L\}$, and let S be the set of indices $i \in [n]$ such that each $c_j \in \mathcal{L}$ has 1 in the ith coordinate. Define x to be the indicator vector of S. Note that $d(x, c_j) = \lambda n - \mathrm{wt}(x) = \lambda n - |S|$, so that $\mathbf{E}\ D_{\mathsf{max}}(x, \mathcal{L}) = \lambda n - \mathbf{E}\ |S|$. Thus to obtain a contradiction, it suffices to show that $\mathbf{E}\ |S| \geqslant \lambda - p = \frac{1}{2}p^L$.

Let $M := |C|$ be the total number of codewords in C, and let M_i be the number of codewords of C with 1 in the i^{th} position. Then the probability that $i \in S$ is equal to $g(M_i)/\binom{M}{L}$, where the function $g : \mathbb{R}^{\geqslant 0} \to \mathbb{R}^{\geqslant 0}$ is defined by $g(z) := \binom{\max\{z, L-1\}}{L}$. By standard closure properties of convex functions, g is convex on $\mathbb{R}$. (Specifically, $z \mapsto \max\{z, L-1\}$ is convex over $\mathbb{R}$, and restricted to its image (i.e., the interval $[L-1, \infty)$), the function $z \mapsto \binom{z}{L}$ is convex. Hence their composition, namely g, is convex as well.)

We are now ready to bound $\mathbf{E}\ |S|$:

$$\frac{1}{n}\mathbf{E}\ |S| \stackrel{(a)}{=} \frac{1}{\binom{M}{L}} \cdot \frac{1}{n}\sum_{i=1}^{n} g(M_i) \stackrel{(b)}{\geqslant} \frac{1}{\binom{M}{L}} \cdot g\left(\frac{1}{n}\sum_{i=1}^{n} M_i\right) = \frac{g(\lambda M)}{\binom{M}{L}} \stackrel{(c)}{=} \frac{\binom{\lambda M}{L}}{\binom{M}{L}}.$$

Here we have used (a) the linearity of expectations, (b) Jensen's inequality, and (c) the fact that $\lambda M \geqslant 2L^2 \geqslant L - 1$. We complete the proof using a straightforward approximation of the binomial coefficients.

$$\mathbf{E}\ |S| \geqslant \frac{(\lambda M - L)^L}{M^L} = \lambda^L \left(1 - \frac{L}{\lambda M}\right)^L \geqslant \lambda^L \left(1 - \frac{L^2}{\lambda M}\right) \geqslant \frac{1}{2}\lambda^L \geqslant \frac{1}{2}p^L.$$

For the second part, by Lemma 2, the rate of a *general* (p, L) list-decodable code is upper bounded by $1 - h\left(p + \frac{1}{2}p^L\right) + o(1)$, which can be shown to be at most $1 - h(p) - \frac{1}{4}(1 - 2p) \cdot p^L + o(1)$. □

The above method can be adapted for q-ary codes with an additional trick:

Theorem 4. *1. Suppose C is a q-ary $(\lambda; p, L)$ list-decodable code with $\lambda = p + \frac{1}{2L}p^L$. Then $|C| \leqslant 2L^2/\lambda$.*
2. Suppose C is a q-ary (p, L) list-decodable code. Then there exists a constant $b = b_{q,p} > 0$ such that the rate of C is at most $1 - h_q(p) - 2^{-bL}$.

Before we provide a proof of this theorem, we will state a convenient lemma due to Erdös. (See Section 2.1 of [13] for reference.) This result was implicitly established in our proof of Theorem 3, so we will omit a formal proof.

Lemma 4 (Erdös 1964). *Suppose $\mathcal{A}$ is a set system over the ground set $[n]$, such that each $A \in \mathcal{A}$ has size at least λn. Then if $|\mathcal{A}| \geqslant 2L^2/\lambda$, then there exist distinct $A_1, A_2, \ldots, A_L$ in $\mathcal{A}$ such that $\bigcap_{i=1}^{L} A_i$ has size at least $\frac{1}{2}n\lambda^L$.*

Proof of Theorem 4: As in Theorem 3, the second part follows from the first. To prove the first claim, assume towards a contradiction that $|C| > 2L^2/\lambda$. Consider the family of sets $\mathcal{A} := \{\mathrm{Supp}(c) : c \in C\}$. By Lemma 4, there exists an L-tuple $\{c_1, c_2, \ldots, c_L\}$ of codewords such that the intersection of their support, say S, has size at least $\frac{1}{2}n\lambda^L \geqslant \frac{1}{2}np^L$. Arbitrarily partition the coordinates in S into L parts, say $S_1, \ldots, S_L$, of almost equal size $\frac{1}{2L}p^L \cdot n$.

Now consider the center x such that x agrees with c_j on all coordinates $i \in S_j$. For $i \notin S$, set x_i to be zero. Then, clearly, $d(x, c_j) \leqslant \lambda n - \frac{1}{2L}p^L \cdot n = pn$. Thus the list $\{c_1, \ldots, c_L\}$ contradicts the (p, L) list-decodability of C. □

5 Constant-Weight vs. General Codes

In this section, we will understand the rate vs. list-size trade-offs for constant-weight codes, that is, codes with every codeword having weight λn, where $\lambda \in (p, \frac{1}{2}]$ is a parameter. (Setting $\lambda = \frac{1}{2}$ roughly corresponds to arbitrary codes having no weight restrictions.) As observed earlier, a typical approach in coding theory to establish rate upper bounds is to study the problem under the above constant-weight restriction. One then proceeds to show a strong negative result of the flavor that a code with the stated properties must have a constant size (and in particular *zero* rate). For instance, the first part of Theorem 3 above is of this form. Finally, mapping this bound to arbitrary codes, one obtains a rate upper bound of $1 - h(\lambda)$ for the original problem. (Note that Lemma 2 provides a particular formal example of the last step.)

In particular, Blinovsky's rate upper bound (Theorem 3) of $1 - h(p) - 2^{-O(L)}$ for (p, L) list-decodable codes follows this approach. (For notational ease, we suppress the dependence on p in the O and Ω notations in this informal discussion.) More precisely, he proves that, under the weight-λ restriction, such code must have zero rate for all $\lambda \leqslant p + 2^{-b_p L}$ for some $b_p < \infty$. One may then imagine improving the rate upper bound to $1 - h(p) - L^{-O(1)}$ *simply by* establishing the latter result for correspondingly higher values of λ (i.e., up to $p + L^{-O(1)}$). We show that this approach cannot work by establishing that (average-radius) list-decodable codes of positive (though possibly small) rates exist as long as $\lambda - p \geqslant 2^{-O(L)}$. Thus Blinovsky's result identifies the correct *zero-rate regime* for the list-decoding problem; in particular, his bound is also the best possible

if we restrict ourselves to this approach. In this context, it is also worth noting that for average-radius list-decodable codes, Theorem 2 already provides a better rate upper bound than what the zero-rate regime suggests, thus indicating that the 'zero-rate regime barrier' is not an inherent obstacle, but more a limitation of the current proof techniques.

In the opposite direction, we show that the task of establishing rate upper bounds for constant weight codes is not significantly harder than the general problem. Formally, we state that that if the "gap to list-decoding capacity" for general codes is γ, then the gap to capacity for weight-λn codes is *at least* $\left(\frac{\lambda-p}{\frac{1}{2}-p}\right)\gamma$. Stated differently, if our goal is to establish a $L^{-O(1)}$ lower bound on the gap γ, then we do not lose by first passing to a suitable λ (that is not too close to p).

5.1 Zero-Rate Regime

Theorem 5. *Fix $p \in (0, \frac{1}{2})$, and set $b = b_p := \frac{1}{2}\left(\frac{1}{2} - p\right)^2$. Then for all sufficiently large L, there exists a $(\lambda; p, L)$ average-radius list-decodable code of rate at least $R - o(1)$, with $p \leqslant \lambda \leqslant p + 5e^{-bL}$ and $R := \min\{e^{-2bL}, e^{-bL}/(6L)\} = \Omega_{p,L}(1)$.*

Proof Sketch: We only provide a sketch of the proof here; see the full version for the complete proof. We obtain this result by random coding followed by expurgation. Set $\varepsilon := e^{-bL}$ and $\lambda' := p + 4\varepsilon$. Consider a random code C of size 2^{Rn} such that each coordinate of each codeword in C is independently set to 1 with probability λ' and to 0 with probability $1 - \lambda'$. For our choice of parameters, we can show that w.h.p., C satisfies the following properties: (a) C is (p, L) average-radius list-decodable; and (b) every codeword in C has weight in the range $(\lambda' \pm \varepsilon)n$. In particular, the maximum weight of any codeword is at most $(p + 5\varepsilon)n$.

Now, pick any C satisfying these two properties, and let C_w denote the subcode of C consisting of the weight-w codewords. Then, if we define $w_0 = \lambda n$ to be the "most popular" weight, then the code C_{w_0} satisfies all our requirements. Note that the final step incurs only a $o(1)$ loss in the rate, since by the pigeonhole principle, the resulting code has size at least $2^{Rn}/(n+1) = \exp_2(Rn - o(n))$. □

5.2 A Reverse Connection between Constant-Weight and Arbitrary Codes

Lemma 5. *Fix p, λ such that $0 < p < \lambda < \frac{1}{2}$. Then in the notation of Definition 3, if $\gamma := 1 - h(p) - R_{p,L}$, then*

$$h(\lambda) - h(p) - \gamma \leqslant R_{p,L}(\lambda) \leqslant h(\lambda) - h(p) - \left(\frac{\lambda - p}{\frac{1}{2} - p}\right)\gamma.$$

Proof: The left inequality is essentially the content of Lemma 2; we show the second inequality here. The manipulations in this proof are of a similar flavor to those in Lemma 3, but the exact details are different.

Suppose C is a $(\lambda; p, L)$ list-decodable code of blocklength n and rate R, such that each codeword in C has weight exactly λn. Pick a random subset $S \subseteq [n]$ of coordinates of size $\alpha_2 n$, with $\alpha_2 := (\lambda - p)/(\frac{1}{2} - p)$, and let $\overline{S} := [n] \setminus S$. (Interestingly, our setting of α_2 differs from the parameter α employed in the proof of Lemma 3 only by a factor of 2. The motivation for this choice of α_2 will become clear shortly.) Consider the subcode C' consisting of codewords $c \in C$ such that $\mathrm{wt}(c|_S) \geqslant \alpha_2 n/2$. For our choice of α_2, one can verify that if $c \in C'$, then c has weight at most $p(1-\alpha_2)n = p|\overline{S}|$ when restricted to $\overline{S}$.

Our key insight now is that the code $C'|_S := \{c|_S \ : \ c \in C'\}$ (of blocklength $\alpha_2 n$) is (p, L) list-decodable. Suppose not. Then there exists a center $x' \in \{0,1\}^S$ and a size-L list $\mathcal{L} \subseteq C$ such that $d(x', c|_S) \leqslant p\alpha_2 n$ for all $c \in \mathcal{L}$. Now, extend x' to $x \in \{0,1\}^n$ such that x agrees with x' on (the coordinates in) S and is zero on the remaining coordinates. Then $\mathcal{L}$ violates the (p, L) list-decodability of C, since for every $c \in \mathcal{L}$,

$$d(x, c) = d(x', c|_S) + \mathrm{wt}(c|_{\overline{S}}) \leqslant p\alpha_2 n + p(1-\alpha_2)n = pn.$$

Hence $C'|_S$ must be (p, L) list-decodable as well. For a fixed $c \in C$, the random variable $\mathrm{wt}(c|_S)$ follows the hypergeometric distribution with parameters $(n, \lambda n, \alpha_2 n)$, which is identical to the hypergeometric distribution with parameters $(n, \alpha_2 n, \lambda n)$. Hence, the probability that c is included in C' is at least

$$\begin{aligned} f(n, \alpha_2 n, \lambda n, \alpha_2 n/2) &= \frac{\binom{\alpha_2 n}{\alpha_2 n/2}\binom{(1-\alpha_2)n}{(\lambda-\alpha_2/2)n}}{\binom{n}{\lambda n}} \\ &\overset{(*)}{=} \frac{\binom{\alpha_2 n}{\alpha_2 n/2}\binom{(1-\alpha_2)n}{p(1-\alpha_2)n}}{\binom{n}{\lambda n}} \\ &\geqslant \exp_2\left(\alpha_2 n + h(p)(1-\alpha_2)n - h(\lambda)n - o(n)\right). \end{aligned}$$

In the step marked $(*)$, we have used the the identity $\lambda - \alpha_2/2 = p(1-\alpha_2)$, which holds for our particular choice of α_2. Thus, summing this over all $c \in C$, the expected size of $C'|_S$ is at least

$$\exp_2\left(Rn + \alpha_2 n + h(p)(1-\alpha_2)n - h(\lambda)n - o(n)\right).$$

On the other hand, since $C'|_S$ is (p, L) list-decodable, the hypothesis of the lemma implies that its size is at most $\exp_2((1 - h(p) - \gamma)\alpha_2 n)$ with probability 1. (It is crucial for our purposes that the blocklength of C' is $\alpha_2 n$, which is significantly smaller than n.) Comparing the upper and lower bound on the expected size of $C'|_S$, we get $R + \alpha_2 + (1-\alpha_2)h(p) - h(\lambda) \leqslant (1 - h(p) - \gamma)\alpha_2$, which can be rearranged to give the desired bound $R \leqslant h(\lambda) - h(p) - \alpha_2\gamma$. □

References

1. Blinovsky, V.M.: Bounds for codes in the case of list decoding of finite volume. Problems of Information Transmission 22(1), 7–19 (1986)
2. Blinovsky, V.M.: Asymptotic Combinatorial Coding Theory. Kluwer Academic Publishers, Boston (1997)

3. Blinovsky, V.M.: Code bounds for multiple packings over a nonbinary finite alphabet. Problems of Information Transmission 41(1), 23–32 (2005)
4. Blinovsky, V.M.: On the convexity of one coding-theory function. Problems of Information Transmission 44(1), 34–39 (2008)
5. Cheraghchi, M., Guruswami, V.: Restricted isometry via list decoding. Work in progress (2012)
6. Elias, P.: List decoding for noisy channels. Technical Report 335, Research Laboratory of Electronics, MIT (1957)
7. Elias, P.: Error-correcting codes for list decoding. IEEE Transactions on Information Theory 37, 5–12 (1991)
8. Guruswami, V.: Linear-algebraic list decoding of folded Reed-Solomon codes. In: Proceedings of the 26th IEEE Conference on Computational Complexity, pp. 77–85 (June 2011)
9. Guruswami, V., Håstad, J., Kopparty, S.: On the list-decodability of random linear codes. IEEE Transactions on Information Theory 57(2), 718–725 (2011)
10. Guruswami, V., Rudra, A.: Explicit codes achieving list decoding capacity: Error-correction up to the Singleton bound. IEEE Transactions on Information Theory 54(1), 135–150 (2008)
11. Guruswami, V., Sudan, M.: Extensions to the Johnson bound (2001) (unpublished manuscript), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.145.9405
12. Guruswami, V., Vadhan, S.P.: A lower bound on list size for list decoding. IEEE Transactions on Information Theory 56(11), 5681–5688 (2010)
13. Jukna, S.: Extremal Combinatorics: with applications in Computer Science. Springer (2001)
14. Rudra, A.: Limits to list decoding of random codes. IEEE Transactions on Information Theory 57(3), 1398–1408 (2011)
15. Wozencraft, J.M.: List Decoding. Quarterly Progress Report, Research Laboratory of Electronics, MIT, vol. 48, pp. 90–95 (1958)
16. Zyablov, V.V., Pinsker, M.S.: List cascade decoding. Problems of Information Transmission 17(4), 29–34 (1981) (in Russian); 236–240 (1982) (in English)

Zero Knowledge LTCs and Their Applications

Yuval Ishai[1], Amit Sahai[2], Michael Viderman[1], and Mor Weiss[1]

[1] Department of Computer Science, Technion, Haifa
{yuvali,viderman,morw}@cs.technion.ac.il
[2] Computer Science Department, UCLA, USA
sahai@cs.ucla.edu

Abstract. Locally testable codes (LTCs) are error-correcting codes for which membership in the code can be tested by probing few symbols of a purported codeword. Motivated by applications in cryptography, we initiate the study of *zero knowledge* locally testable codes (ZK-LTCs). ZK-LTCs are LTCs which admit a randomized encoding function, such that even a malicious tester which reads a large number of codeword symbols learns essentially nothing about the encoded message.

We obtain ZK-LTCs with good parameters by applying general transformations to standard LTCs. We also obtain LTCs and ZK-LTCs which are *stable* in the sense that they limit the influence of adaptively corrupted symbols on the output of the testing procedure.

Finally, we apply stable ZK-LTCs for obtaining protocols for verifiable secret sharing (VSS) in which the communication complexity required for verifying a shared secret is sublinear in the secrecy threshold. We also obtain the first statistically secure VSS protocols and distributed coin-flipping protocols which use n servers, tolerate a constant fraction of corrupted servers, and have error that vanishes almost exponentially with n using only $O(n)$ bits of communication. These improve over previous VSS and coin-flipping protocols from the literature, which require nearly quadratic communication to achieve similar guarantees.

1 Introduction

In this work we initiate the study of locally testable codes (LTCs) with a zero knowledge property and apply such codes towards the design of efficient cryptographic protocols. Before describing the questions we consider and our main results, we first give a short overview of LTCs.

Locally Testable Codes. An LTC is a code with a *tester*, a randomized algorithm with oracle access to a received word w. The tester reads few symbols of w, and based on this "local view" decides whether w is in the code. The tester should have perfect *completeness*, i.e., it should accept codewords with probability one, and a small *soundness error*, namely it should reject with high probability words that are far from the code.[1] Implicit already in [2] (cf. [21,

[1] There are two kinds of LTCs: weak and strong. In a weak LTC the tester is only guaranteed to distinguish between a valid codeword and w which is far from the code, whereas in a strong LTC the tester's rejection probability is proportional to the distance from the code. We refer the reader to Section 2 for formal definitions.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 607–622, 2013.

Sec. 2.4]), LTCs were first explicitly studied by Goldreich and Sudan [23]. LTCs are of interest in computer science due to their connections to probabilistically checkable proofs (PCPs) and property testing (see surveys [32,21] for details).

Our work is motivated by the possibility of applying the efficient verification feature of LTCs in the contexts of distributed storage and cryptographic protocols. Suppose that a user wishes to reliably store her data by distributing it among a large number of potentially unreliable, or even malicious, servers. To ensure the integrity of the data, the user can apply a good error correcting code before distributing the data. However, traditional codes lack two useful properties. First, they do not admit an efficient procedure for checking whether the stored data has been tampered with to an extent which may compromise its integrity. Second, malicious servers may gather a significant amount of information about the data, which is problematic when the data is sensitive.

The above scenario naturally gives rise to the notion of *zero knowledge* LTCs. A zero knowledge LTC is an LTC equipped with a randomized encoding function which ensures that the encoded data remain hidden, even in the presence of a coalition of malicious servers which may observe a bounded number of symbols in the encoding. Note, however, that in the above scenario the standard notion of testing may be insufficient, since malicious servers may adaptively determine their answers *after* seeing the queries of the user. This calls for a notion of *stability* - a stronger form of testability which we introduce and study. A more detailed discussion of the stability and zero knowledge properties of LTCs follows.

Stable LTCs. As noted above, the standard notion of LTCs does not consider the case in which an adversary can adaptively control some symbols in a purported codeword, based on the locations of the symbols that the tester queries. We will be interested in *stable LTCs* which tolerate such adversaries. More concretely, stability concerns the situation in which a tester $\mathcal{D}$ for the code $C \subseteq \mathbb{F}^n$ tests whether $w \in \mathbb{F}^n$ is in C. The testing is performed in the presence of an adversary $\mathcal{A}$, who selects in advance a subset $T \subseteq [n]$ of size t and can adaptively modify the symbols indexed by T after seeing the queries of $\mathcal{D}$. (In particular, the answers to the queries of $\mathcal{D}$ are according to w and the symbols that $\mathcal{A}$ chose to alter.) The goal of $\mathcal{A}$ is to cause $\mathcal{D}$ to reject when $w \in C$, or alternatively to accept when w is far from C. Informally, a code C is a (t, ϵ, δ)-stable LTC with respect to the tester $\mathcal{D}$, if $\mathcal{D}$ can distinguish (except with at most ϵ error) between codewords in C and words that are at least δ-far from C, even in the presence of an adversary $\mathcal{A}$ as above.

ZK-LTCs. In this paper we initiate the study of zero knowledge LTCs (ZK-LTCs). We begin by describing a more general notion of ZK-codes.[2] Informally, we say that a code C with a *randomized* encoding function E_C is t-ZK if an adversary who reads at most t symbols of a codeword $c = E_C(x)$ learns no information about the message x. We say that E_C it is (t, ϵ)-ZK if the advantage of such an adversary in distinguishing between two messages is bounded by ϵ.

[2] Such codes were studied by Feldman et al. [17] under the name "secure coding schemes".

A ZK-LTC is an LTC with a ZK encoding function. We note that for cryptographic applications of ZK-LTCs, we will typically be interested in codes with $t = \Omega(n)$ where the honest tester needs to probe only a sublinear number of symbols. Such ZK-LTCs will be used to obtain efficient cryptographic protocols tolerating a constant fraction of corrupted parties.

1.1 Our Results

Our contributions are three-fold. First, we construct efficient ZK-LTCs by applying and extending previous general transformations from codes to ZK-codes. More specifically, using a probabilistic transformation of linear codes into ZK-codes due to Feldman et al. [17], we obtain (Corollary 1) constant rate ZK-LTCs with sublinear query complexity and a constant fractional ZK parameter. We also give a fully explicit general transformation from linear codes to ZK-codes (Theorem 2) as well as a non-explicit transformation of arbitrary codes to ZK-codes (Theorem 3). Second, we construct good LTCs and ZK-LTCs with the additional stability guarantee (Theorem 4). Finally, we demonstrate the usefulness of stable ZK-LTCs by applying them towards the design of efficient protocols for verifiable secret sharing (Theorems 6 and 5) and distributed coin-flipping (Theorem 7). We now give a more detailed account of our results.

Constructing ZK-LTCs. Feldman at al. [17, Section 3.3] gave a probabilistic construction of ZK-codes with good parameters from linear codes. In Theorem 1 we slightly generalize their result, and formulate a common requirement which is valid for both large and small fields. Then, in Corollary 1 we state the parameters of ZK-LTCs that can be obtained by applying Theorem 1 to LTCs from the literature.

Theorem 1 (Linear codes to ZK-codes, probabilistically). *Let $C \subseteq \mathbb{F}^n$ be a linear code of dimension k and let $k' < k$ be a positive integer. Assume that for some $0 < t \leq \frac{n}{2}$ we have*

$$\log(|\mathbb{F}|) \cdot (k - k' - t) > t \cdot \log\left(\frac{n}{t}\right) + 4t.$$

Then there exists a generator matrix G' for C such that C, under the randomized encoding defined by $E_C(x; r) = G' \cdot (x; r)$, is a t-ZK code for messages x in $\mathbb{F}^{k'}$. Furthermore, such a G' can be constructed in probabilistic polynomial time (except with negligible failure probability) given k', t, and any generator matrix for C.

The following corollary states the existence of asymptotically good strong LTCs with a constant fractional ZK parameter t. We use $\text{rate}(E_C)$ to denote the rate of the randomized encoding function E_C, i.e., the ratio between the length of the original message and the length of the encoded message. See Section 2 for the definition of strong LTCs.

Corollary 1 (Probabilistic construction of ZK-LTCs). *For every $0 < \gamma < 1/2$ there is a finite field $\mathbb{F}$ of constant size (depending only on γ) such that for every $\lambda > 0$ the following holds. For every positive integer k', there is a linear code $C \subseteq \mathbb{F}^n$ of dimension k and a generator matrix G' for C, such that:*

- *C is an $(n^\lambda, \frac{1}{2})$-strong LTC with relative distance $\delta(C) = \Omega_\lambda(1)$.*
- *The randomized encoding $E_C(x; r) = G' \cdot (x; r)$, where $x \in \mathbb{F}^{k'}$ and $r \in \mathbb{F}^{k-k'}$, satisfies $\mathrm{rate}(E_C) = k'/n = \Omega_\lambda(1)$ and is t-ZK for $t = \lfloor \gamma n \rfloor$.*

Furthermore, given k', such a G' can be constructed in probabilistic poly(k') time (except with negligible failure probability).

Due to space limitations, we defer the proofs of Theorem 1 and Corollary 1, as well as some extensions, to the full version.

The main drawback of the result of [17] is that the ZK-encoding is constructed probabilistically. This means that applications which use such an encoding may need to rely on a trusted setup in which the generator matrix G' describing the encoding E_C is picked at random. To avoid such a setup, we present a fully explicit transformation of linear codes to ZK-codes. The main price we pay is that the zero knowledge property of the encoding becomes statistical rather than perfect. The idea is to first encode the message by a randomized encoding which has the property that any linear function $L : \mathbb{F}^k \to \mathbb{F}$ has only a small statistical advantage in distinguishing between encodings of two different messages, and then apply the given linear encoding to the result. This approach yields the following theorem (see Section 3).

Theorem 2 (Linear codes to ZK-codes, explicitly). *Let $\mathbb{F} = \mathbb{F}_2$ be the binary field. For any linear code $C \subseteq \mathbb{F}^n$ of dimension k, there exists a randomized encoding function $E_C : \mathbb{F}^{k'} \to \mathbb{F}^n$, whose image is contained in C, such that $k' = \Omega(k)$ and E_C is (t, ϵ)-ZK for $t = \Omega(k)$ and $\epsilon = 2^{-\Omega(k)}$. Furthermore, a randomized circuit for computing E_C (resp., a circuit for inverting E_C) can be computed in deterministic polynomial time given any generator matrix for C.*

While all of the LTCs we will rely on are *linear*, there could potentially be non-linear LTCs with better or incomparable parameters (an example being the Long Code used in efficient PCP constructions, cf. [14]). We observe that for a general (not necessarily linear) code C, a completely random choice of an encoding function has good ZK parameters with high probability. This is captured by Theorem 3 below. Note that this construction is less explicit than the one given by Theorem 1 in that the encoding function is inefficient.

Theorem 3 (General codes to ZK-codes, non-explicitly). *Let $\mathbb{F} = \mathbb{F}_2$ be the binary field. For any code $C \subseteq \mathbb{F}^n$ with $|C| = 2^k$, there exists a randomized encoding function $E_C : \mathbb{F}^{k'} \to \mathbb{F}^n$, whose image is contained in C, such that $k' = \Omega(k)$ and E_C is (t, ϵ)-ZK for $t = \Omega(k)$ and $\epsilon = 2^{-\Omega(k)}$.*

The proof of Theorem 3 is deferred to the full version.

Stable LTCs. In Section 4 we construct a good family of stable linear LTCs. These are transformed into stable ZK-LTCs, which are later used in cryptographic protocols. The parameters of the stable ZK-LTCs we obtain are summarized by Theorem 4 below.

Our construction of stable LTCs is based on the testability of the tensor products of codes studied in [15,9,8,7,34]. Intuitively, the codewords in such codes are multidimensional tensors, whose restriction to every axis-parallel line belongs to the fixed base code. Given an input word, a tester picks a random 2-dimensional hyperplane and reads all symbols of the input word indexed by this hyperplane coordinates. Based on this "local view", the tester decides whether to accept or reject the input word. It was shown [7,34] that such codes are linear strong LTCs.

Our contribution here is in showing that tensor products of codes can yield stable LTCs. Concretely, we show that if a base code is an efficiently decodable linear code, then its tensor products are stable LTCs with respect to a new tester. The proof of Theorem 4 is based on the robust testability of tensor products [34] and the efficiently decodable linear codes from [31].

Theorem 4 (Stable ZK-LTCs). *There exists a constant $\alpha > 0$ such that for any constants $\lambda > 0$ and $\delta \in (0, \alpha)$, there is a family of binary linear codes $C \subseteq \mathbb{F}_2^n$ with a linear randomized encoding function E_C, such that:*

- *C is efficiently decodable from αn errors.*
- *C is a $(t, \frac{1}{4}, \delta)$-stable LTC with respect to a $\mathrm{poly}(n)$-time tester making n^λ queries and using $O(\log(n))$ random bits.*
- *E_C is t-ZK for $t = \Omega_\lambda(n)$.*
- *$\mathrm{rate}(E_C) = \Omega_\lambda(1)$.*

1.2 Cryptographic Applications

We use stable ZK-LTCs to design efficient protocols for verifiable secret sharing and distributed coin-flipping. In both cases, we assume that the participating parties can interact over a synchronous network of secure point-to-point channels. The parties also have access to a *broadcast* channel, where a message sent over this channel is received by all other parties. When measuring communication complexity, we count a message sent over a broadcast channel only once towards the total communication. Alternatively, our protocols can be implemented with similar communication complexity using a public bulletin board, where every time a message is written to or read from the board is counted towards the communication complexity.

The security of protocols is defined by considering their execution in the presence of an active adversary who may corrupt and control a subset of the parties. We assume adversaries to be *static*, in the sense that they choose the set of corrupted parties at the onset of the protocol, but they are capable of *rushing*, namely sending their messages only after receiving all messages sent by honest parties in the same round.

Verifiable Secret Sharing. Verifiable secrete sharing (VSS) [11,18,3,10,4] is a standard building block for cryptographic multi-party protocols. In a nutshell, a VSS protocol allows a dealer D to distribute a secret s among n servers in a way that prevents a coalition of up to t servers from learning or modifying the secret, while on the other hand guaranteeing unique reconstruction even if D and up to t servers can collude.

We consider the following *designated receiver* variant of VSS which involves, in addition to D and the n servers, a designated receiver R who may assist in the verification. Such a VSS protocol consists of three phases. In the *sharing phase*, the dealer D randomly distributes s between the servers by privately sending a share s_i to each server. In the *verification phase*, the receiver R can freely interact with the servers, possibly using a broadcast channel. Finally, in the *reconstruction phase*, each server sends a single message to R, and R reconstructs the secret.

A protocol as above is said to be (t, ϵ)-secure if it satisfies the following requirements:

- *Correctness.* If the adversary corrupts t servers, the receiver reconstructs the correct secret s except with at most ϵ probability.
- *Secrecy.* Any adversary who corrupts R and t servers gets at most an ϵ-advantage in distinguishing between two secrets.
- *Binding.* For any adversary who corrupts D and t servers, the following holds except with at most ϵ failure probability over the randomness of the sharing and verification phases. In the end of the verification phase there is a unique secret s^* (determined by the messages exchanged up to this point), such that R will output s^* regardless of the messages sent by the adversary during the reconstruction phase.

It is instructive to note that if the binding requirement is relaxed so that R is only guaranteed to either output s^* or reject (in a way that may depend on the adversary's messages during reconstruction) then the problem becomes much easier to solve [4,19]. This weaker variant, sometimes referred to as *weak VSS* or *distributed commitment*, does not suffice for several applications of VSS including the coin-flipping protocols we present next. On the other hand, traditional VSS is stronger than our designated receiver variant in that the verification phase does not involve the receiver R. Thus, when there are multiple receivers, traditional VSS can guarantee that the same secret s^* be reconstructed by all receivers, whereas applying our VSS verification with each receiver separately does not. Still, designated receiver VSS is as good as traditional VSS in situations where agreement between different receivers is not required, as in the coin-flipping application we describe next. We are not aware of any simpler or better solutions to the problem of designated receiver VSS using previous VSS techniques from the literature.

From ZK-LTC to VSS. The application of ZK-LTCs to VSS is conceptually simple. The protocol uses a *stable* ZK-LTC C with encoding function E_C and tester $\mathcal{D}$. (For the protocol to be computationally efficient, we need E_C to be

efficiently encodable and decodable.) In the sharing phase, the dealer D randomly encodes the secret s into a codeword $c = E_C(s) \in C$, and partitions the symbols of c between the servers. In the verification phase, the receiver R verifies that the messages received by the servers are close to a valid codeword by executing a random test of $\mathcal{D}$, where R broadcasts the queries of $\mathcal{D}$ and the servers answer. (The use of broadcast prevents R from contacting too many servers, which would violate the secrecy requirement.) If the test fails, R outputs $s^* = 0$ (or some other default value) and ignores further messages. For reconstruction, the servers send their shares to R, who decodes the secret s. The zero knowledge property of E_C implies the secrecy property. The stability of $\mathcal{D}$ is useful for both binding and correctness. First, it ensures that if a corrupted D distributes c^* which is far from C then R notices this with high probability, even if D and up to t servers collude. Second, if D is honest and $c \in C$, the tester $\mathcal{D}$ will accept and the decoder of E_C will output $s^* = s$ in the end of the reconstruction phase (except with small probability) even if there are t malicious servers. Finally, if a corrupted D distributes c^* which is not a codeword but is not far from C, then regardless of whether $\mathcal{D}$ accepts or rejects a unique secret s^* is determined in the end of the verification phase.

Sublinear verification. Note that by using good stable ZK-LTCs, the communication complexity in the verification phase of the above protocol becomes sublinear in the secrecy threshold t. Sublinear verification can be motivated by situations in which verification forms an efficiency bottleneck. Consider, for example, a situation in which the secret is reconstructed long after the shares are distributed. As more and more servers may become corrupted over time, the receiver might want to run the verification procedure periodically, whereas the shares are distributed (and the secret is reconstructed) only once. The efficient verification feature is captured by the following theorem.

Theorem 5. *For every constant $\lambda > 0$ there exists a constant-round, n-server, designated receiver VSS protocol for secrets of length $\Omega(n)$ with verification phase which uses $O(n^\lambda)$ bits of communication. The protocol is (t, ϵ)-secure for $t = \Omega(n)$ and $\epsilon = n^{-\omega(1)}$.*

Linear total communication. Another feature of the designated receiver VSS protocols we obtain via stable ZK-LTCs is that they can be implemented with only $O(n)$ bits of communication, $t = \Omega(n)$ and statistical error ϵ that vanishes almost exponentially with n. Previous VSS protocols (e.g., those from [3,10,4,16,20,13,19,29,27,30,1]) require nearly quadratic communication (or more) to achieve similar guarantees even when the secret is just a single bit, though they can offer a higher fractional resilience and are not restricted to a designated receiver. The linear communication feature is captured by the following theorem.

Theorem 6. *For every constant $\lambda > 0$ there exists a constant-round, n-server, designated receiver VSS protocol for secrets of length $\Omega(n)$ with total communication complexity $O(n)$. The protocol is (t, ϵ)-secure for $t = \Omega(n)$ and $\epsilon = 2^{-n^{1-\lambda}}$.*

Distributed Coin-Flipping. We consider a distributed model for coin-flipping in which two clients want to agree on a common random bit with the help of n servers. The clients can interact with the servers via synchronous, secure point-to-point channels and a broadcast channel, where at the end of the interaction each client outputs a single bit. The protocol is said to be (t, ϵ)-secure if the following requirements are met.

- *Correctness.* If an adversary can corrupt t servers at the onset of the protocol, the joint outputs of the two clients will be ϵ-close (in statistical distance) to a pair of identical random bits.
- *Agreement.* An adversary who corrupts one client and t servers at the onset of the protocol can bias the output of the other client by at most ϵ.

The above distributed model for coin-flipping is motivated by the impossibility of achieving a similar fairness guarantee via direct interaction between the clients. This impossibility holds even if one settles for security against computationally bounded clients [12].

Our coin-flipping protocols are obtained from designated-receiver VSS in the natural way: each client picks a random $s \in \{0, 1\}$ and distributes s between the servers using the VSS protocol, where the other client acts as the receiver. The two secret bits are then reconstructed, and the common coin is defined as their exclusive-or. The communication complexity of the protocols obtained via this approach is captured by the following theorem.

Theorem 7. *For every constant $\lambda > 0$ there exists a constant-round n-server (t, ϵ)-secure distributed coin-flipping protocol, where $t = \Omega(n)$ and $\epsilon = 2^{-n^{1-\lambda}}$, with total communication complexity $O(n)$.*

As before, coin-flipping protocols which are based on VSS protocols from the literature require nearly quadratic communication to achieve a similar security guarantee.

Related Work. The notion of *zero-knowledge PCPs,* a PCP analogue of ZK-LTCs, was studied in [26,25]. Zero-knowledge PCPs do not seem to imply good ZK-LTCs or any of the applications presented in this work, and their construction is considerably more involved.

2 Preliminaries

An error correcting code over the alphabet Σ is a subset $C \subseteq \Sigma^n$. The code can also be associated with an injective encoding function E_C that maps a set of messages Σ^k to a set of codewords, i.e., $C = \left\{E_C(x) \mid x \in \Sigma^k\right\}$. We will also consider randomized encoding functions E_C, which map messages from $\Sigma^{k'}$ for some $k' < k$ into codewords of C. We assume that such a randomized encoding is injective, in that the codeword distributions associated with different messages have disjoint support sets.

Most of the well-studied and practically used codes are linear codes. A linear code $C \subseteq \mathbb{F}^n$ is a linear subspace over the field $\mathbb{F}$, where n is called the blocklength of C and $\dim(C)$ denotes the dimension of the code. The rate of the code is defined by $\text{rate}(C) = \frac{\dim(C)}{n}$. If C is linear then, letting $k = \dim(C)$, an encoding function for C can be associated with a generator matrix $G \in \mathbb{F}^{n \times k}$, namely the encoding is done by multiplying G on the message vector such that $C = \left\{G \cdot x \mid x \in \mathbb{F}^k\right\}$.

We define the distance between two words $x, y \in \mathbb{F}^n$ to be $\Delta(x, y) = |\{i \mid x_i \neq y_i\}|$, and the relative distance to be $\delta(x, y) = \frac{\Delta(x,y)}{n}$. The distance of the code C is defined as $\Delta(C) = \min_{x \neq y \in C} \Delta(x, y)$ and its relative distance is denoted $\delta(C) = \frac{\Delta(C)}{n}$. Typically, one is interested in codes whose distance is linear in the blocklength.

For $w \in \mathbb{F}^n$, let $\text{supp}(w) = \{i \in [n] \mid w_i \neq 0\}$. For a word $x \in \mathbb{F}^n$ and a code $C \subseteq \mathbb{F}^n$ we let $\delta(x, C) = \min\{\delta(x, c) \mid c \in C\}$. Given $\delta \in [0, 1]$, we say that $x \in \mathbb{F}^n$ is δ-*far* from $C \subseteq \mathbb{F}^n$ if $\delta(x, C) \geq \delta$, otherwise x is δ-*close* to C.

2.1 Efficiently Encodable and Decodable Codes

When considering an infinite family of codes $C \subseteq \mathbb{F}^n$ with varying block length together with associated encoding functions $E_C : \mathbb{F}^k \to \mathbb{F}^n$, we say that C is *efficiently encodable* if given a message $x \in \mathbb{F}^k$, the codeword $E_C(x)$ can be computed in polynomial time. Notice that if C is linear then it is always encodable in time $O(k \cdot n) = O(n^2)$ given its generator matrix.

We say that $C \subseteq \mathbb{F}^n$ is *efficiently decodable* from $l < \Delta(C)/2$ errors, if there exists a poly(n)-time *decoding algorithm* Dec that on any input $w \in \mathbb{F}^n$ such that $\Delta(w, C) \leq l$, outputs a codeword $c \in C$ such that $\Delta(w, c) \leq l$. (Namely, the decoding algorithm outputs the closest codeword.)

Sometimes, instead of finding the closest codeword, we will need to obtain the original message, i.e., a message $m \in \mathbb{F}^k$ such that $\Delta(w, E_C(m)) \leq l$. We note that when C is linear, this task is equivalent to "standard" decoding, since after obtaining the closest codeword $c \in C$, matrix multiplication can be used to easily find an $m \in \mathbb{F}^k$ such that $E_C(m) = c$. Thus, if a linear code C is efficiently decodable from l errors, then the the encoded message can also be found efficiently, even if l errors occurred.

2.2 Locally Testable Codes

Let $[n]$ denote the set $\{1, \ldots, n\}$. For $w \in \mathbb{F}^n$ and $S \subseteq [n]$, let $w|_S$ be the restriction of w to the subset S. Similarly, let $C|_S = \{c|_S \mid c \in C\}$ denote the projection of the code C onto S.

A *standard q-query tester* for a linear code $C \subseteq \mathbb{F}^n$ is a randomized algorithm that on the input word $w \in \mathbb{F}^n$ picks non-adaptively a subset $I \subseteq [n]$ such that $|I| \leq q$. Then the tester reads all symbols of $w|_I$, accepts if $w|_I \in C|_I$, and rejects otherwise (see [6, Theorem 2]). Hence a q-query tester can be associated with a distribution $\mathcal{D}$ over subsets $I \subseteq [n]$ such that $|I| \leq q$.

Definition 1 (Testers and LTCs). *A q-query tester for a linear code $C \subseteq \mathbb{F}^n$ is a distribution $\mathcal{D}$ over subsets $I \subseteq [n]$ such that $|I| \leq q$.*

- *A q-query tester $\mathcal{D}$ is a (q, ϵ, ρ)-tester if for all $w \in \mathbb{F}^n$ such that $\delta(w, C) \geq \rho$, $\Pr_{I \sim \mathcal{D}}[w|_I \notin C|_I] \geq \epsilon$.*
- *A q-query tester $\mathcal{D}$ is a (q, ϵ)-strong tester if for all $w \in \mathbb{F}^n$, $\Pr_{I \sim \mathcal{D}}[w|_I \notin \mathcal{C}|_I] \geq \epsilon \cdot \delta(w, C)$.*

A code $C \subseteq \mathbb{F}^n$ is a (q, ϵ, ρ)-weak LTC *if it has a (q, ϵ, ρ)-tester, and it is a* (q, ϵ)-strong LTC *if it has a (q, ϵ)-strong tester.*

Although the tester in Definition 1 does not output accept or reject as a standard tester does, it can be converted to output accept, reject based on its local view $w|_I$.

Stable LTCs. As noted above, standard correctness and soundness are insufficient for cryptographic applications of LTCs, since standard testing is defined only in relation to a *pre-determined* purported codeword. In our setting, on the other hand, a t-adversary (namely, an adversary that can alter at most t symbols of the word) can *adaptively* change few coordinates of the word *after* seeing the queries of the tester. More specifically, we consider two possible scenarios; first, that the adversary tries to cause the tester to reject codewords, and second that he tries to cause the tester to accept words that are far from the code.

The main difference from standard testing is as follows. First of all, the t-adversary $\mathcal{A}$ selects a subset $T \subseteq [n], |T| = t$. Next, the tester $\mathcal{D}$ picks (non-adaptively) a subset of coordinates $I \subseteq [n]$ to query. Then, $\mathcal{A}$ is given I, and can modify any symbol of w indexed by T, obtaining a $w' \in \mathbb{F}^n$ such that $\mathrm{supp}(w' - w) \subseteq T$. Finally, the tester receives $w'|_I$ (i.e., the queried symbols after the above modifications), and either accepts or rejects.

Definition 2 (Stable LTCs). *Let $C \subseteq \mathbb{F}^n$ be a code with tester $\mathcal{D}$.*

- *(Stable completeness) $\mathcal{D}$ has (t, ϵ)-completeness, if for every t-adversary $\mathcal{A}$ and every codeword $c \in C$, the tester $\mathcal{D}$ rejects c with probability at most ϵ.*
- *(Stable soundness) $\mathcal{D}$ has (t, ϵ, δ)-soundness, if for every t-adversary $\mathcal{A}$ and every word $w \in \mathbb{F}^n$ such that $\delta(w, C) \geq \delta$, the tester $\mathcal{D}$ accepts w with probability at most ϵ.*

We say that C is a (t, ϵ, δ)-stable LTC if it has (t, ϵ)-completeness and (t, ϵ, δ)-soundness.

Zero Knowledge Codes. We consider both perfect and statistical zero knowledge. Informally, a randomized encoding function E_C is t-ZK, if any adversary reading t symbols from an encoded message learns nothing about the message. A standard (deterministic) encoding cannot even be 1-ZK, since there must be a symbol of the output which depends on the message. In the case of a linear code $C \subseteq \mathbb{F}^n$ of dimension k, any generator matrix G' for C together with a

message length parameter $k' < k$ define a randomized encoding E_C mapping $x \in \mathbb{F}^{k'}$ to $E_C(x;r) = G'(x;r)$, where $r \in \mathbb{F}^{k-k'}$. We define the rate of E_C as $\text{rate}(E_C) = \frac{k'}{n}$. More generally, E_C can be an arbitrary injective randomized mapping from $\mathbb{F}^{k'}$ to C. We now define the zero knowledge property of E_C. We let $\mathsf{SD}(X,Y)$ denote the statistical distance between the distributions X and Y.

Definition 3 (ZK-codes). *Let n, k' be positive integers. Let $C \subseteq \mathbb{F}^n$ be a code and let $E_C : \mathbb{F}^{k'} \to \mathbb{F}^n$ be an associated randomized encoding function. We say that E_C is (t,ϵ)-ZK, if for every set $I \subseteq [n]$ of size t and every message pair $x, x' \in \mathbb{F}^{k'}$,*

$$\mathsf{SD}\left(E_C(x)|_I\,,\, E_C(x')|_I\right) \leq \epsilon.$$

We say that E_C is t-ZK if it is $(t,0)$-ZK.

3 Explicit ZK-Codes

In this section we prove the explicit transformation of linear codes into ZK-codes stated in Theorem 2. We rely on the following lemma, which generalizes Vazirani's XOR Lemma [33,22] and can be proved similarly.

Lemma 1 (XOR lemma). *Let X and Y be distributions over $\mathbb{F}_2^k$ such that $\mathsf{SD}(X,Y) = \epsilon$. Then there exists an $\alpha \in \mathbb{F}_2^k$ such that*

$$\mathsf{SD}(\alpha^T X\,,\, \alpha^T Y) \geq \epsilon/2^{k/2}.$$

We will use randomized encoding functions that fool linear distinguishers. This is captured by the following definition.

Definition 4 (Linear-secure encoding). *Let $\mathbb{F}$ be a finite field. A randomized encoding $\text{Enc} : \mathbb{F}^{k'} \to \mathbb{F}^k$ is ϵ-secure against linear distinguishers if for every $x, x' \in \mathbb{F}^{k'}$ and for every linear function $L : \mathbb{F}^k \to \mathbb{F}$ we have*

$$\mathsf{SD}\left(L(\text{Enc}(x))\,,\, L(\text{Enc}(x'))\right) \leq \epsilon.$$

The following is implicit in [24].

Lemma 2 (Constant-rate linear-secure encoding). *Let $\mathbb{F} = \mathbb{F}_2$ be the binary field. There exists a probabilistic polynomial-time algorithm computing a randomized encoding $\text{Enc} : \mathbb{F}^{k'} \to \mathbb{F}^{k(k')}$ such that $k(k') = O(k')$ and Enc is ϵ-secure against linear distinguishers for $\epsilon(k') = 2^{-\Omega(k')}$. Furthermore, there exists a polynomial-time algorithm Dec such that $\Pr[\text{Dec}(\text{Enc}(x)) = x] = 1$ for all x.*

Applying a linear encoding function on top of a linear-secure encoding with sufficiently small ϵ yields a good (t,ϵ)-ZK code.

Lemma 3. *Let $\mathbb{F}$ be a finite field. Let $C \subseteq \mathbb{F}^n$ be a linear code with $\dim(C) = k$ and let G be a generator matrix for C. Let $\text{Enc} : \mathbb{F}^{k'} \to \mathbb{F}^k$ be a randomized encoding such that Enc is ϵ-secure against linear functions. Then, for every positive integer t, the randomized encoding $E_C : \mathbb{F}^{k'} \to \mathbb{F}^n$ defined by $E_C(x) = G \cdot \text{Enc}(x)$ is $\left(t, 2^{t/2} \cdot \epsilon\right)$-ZK.*

Proof. Suppose towards contradiction that there are $x, x' \in \mathbb{F}^{k'}$ and a set $I \subseteq [n]$ of size t such that $\mathsf{SD}(E_C(x)|_I, E_C(x')|_I) > 2^{t/2} \cdot \epsilon$. By Lemma 1 there exists $\alpha \in \mathbb{F}^t$ such that $\mathsf{SD}(\alpha^T \cdot E_C(x)|_I, \alpha^T \cdot E_C(x')|_I) > \epsilon$. It follows that there is $\beta \in \mathbb{F}^n$ such that $\mathsf{SD}(\beta^T G \cdot \mathrm{Enc}(x), \beta^T G \cdot \mathrm{Enc}(x')) > \epsilon$, contradicting the ϵ-security of Enc. □

Theorem 2 follows by applying Lemma 3 to the efficient linear-secure encoding guaranteed by Lemma 2.

4 Stable LTCs via Tensor Product Codes

In this section we prove Theorem 4, thus showing the existence of stable LTCs with sufficiently nice parameters. We define tensor products in Section 4.1 and prove Theorem 4 in Section 4.2.

4.1 Tensor Product of Codes

The definitions presented here are standard in the literature on tensor-based LTCs (e.g., [15,7,28,9]).

Let I and J be sets of coordinates. For $x \in \mathbb{F}^J$ and $y \in \mathbb{F}^I$ we let $x \otimes y$ denote the tensor product of x and y (i.e., the matrix M with entries $M_{(i,j)} = x_j \cdot y_i$ where $(i,j) \in I \times J$). Let $R \subseteq \mathbb{F}^J$ and $C \subseteq \mathbb{F}^I$ be linear codes. We define the tensor product code $R \otimes C$ to be the linear space spanned by words $r \otimes c \in \mathbb{F}^{I \times J}$ for $r \in R$ and $c \in C$.

We let $C^1 = C$ and $C^t = C^{t-1} \otimes C$ for $t > 1$. Note that for this definition, $C^{2^0} = C$ and $C^{2^t} = C^{2^{t-1}} \otimes C^{2^{t-1}}$ for $t > 0$. We also note that for a code $C \subseteq \mathbb{F}^n$ and $m \geq 1$ it holds that $\mathrm{rate}(C^m) = (\mathrm{rate}(C))^m$, $\delta(C^m) = (\delta(C))^m$ and the blocklength of C^m is n^m.

Testers for C^m. Let us present the testers for these codes. A point in an m-dimensional cube can be associated with an m-tuple $(i_1, i_2, ..., i_m)$ such that $i_j \in [n]$. We say that τ is an $(m-1)$-dimensional (b,i)-hyperplane (or simply a hyperplane, in short) if

$$\tau = \{(i_1, i_2, ..., i_m) \mid i_b = i \text{ and for all } j \in [m] \setminus \{b\} \text{ we have } i_j \in [n]\}.$$

Definition 5 (Hyperplane Tester). *Let $m \geq 3$. Let $M \in \mathbb{F}^{n^m}$ be an input word and think of testing whether $M \in C^m$. The $(m-1)$-dimensional hyperplane tester $\mathcal{D}$ picks (non-adaptively) a random $b \in [m]$ and a random $i \in [n]$, and returns the (b,i)-hyperplane (the corresponding local view is $M|_{(b,i)}$). It is not hard to prove that if $M \in C^m$ then $M|_{(b,i)} \in C^{m-1}$.*

Such testers can be composed to yield the following tester. Given a candidate word $M \in \mathbb{F}_2^{n^m}$, the tester picks a random $(m-1)$-dimensional hyperplane τ, and considers $M' = M|_\tau$ which is a candidate to be in C^{m-1}. Next, the tester can pick a random $(m-2)$-dimensional hyperplane τ_1 and considers $M'' = M'|_{\tau_1}$, etc. We define a tester with query complexity n^2 by picking a random 2-dimensional hyperplane when the blocklength of the code C^m is n^m.

Definition 6 (2-dimensional hyperplane tester). *Let $m \geq 3$. Let $M \in \mathbb{F}^{n^m}$ be an input word and think of testing whether $M \in C^m$. The 2-dimensional hyperplane tester $\mathcal{D}$ picks random $b_1, b_2, \ldots, b_{m-2} \in [m]$ such that $b_{i_1} \neq b_{i_2}$ for $i_1 \neq i_2$ and random $j_{b_1}, j_{b_2}, \ldots, j_{b_{m-2}} \in [n]$, and outputs $M|_\tau$, where $B = \{b_1, b_2, \ldots, b_{m-2}\}$ and $\tau = \{(i_1, i_2, \ldots, i_m) \in [n]^m \mid \forall g \in B : i_g = j_g\}$. Note that for any such choice we have $C^m|_\tau = C^2$.*

Remark 1. We notice that a 2-dimensional hyperplane tester for the code C^m uses at most $\log\left(\binom{m}{2} \cdot n^{(m-2)}\right) \leq 2m\log(n)$ random bits.

Robust Testing. We now define the notion of *robustness* (Definition 7) as was introduced in [7]. Informally, we say that a tester is robust if for every word that is far from the code, the tester's view is far on average from any consistent view. (This notion was defined for LTCs following an analogous definition for PCPs [5,14].) Formally,

Definition 7 (Robustness). *Let $C \subseteq \mathbb{F}^n$ be a code. Note that $\Delta(w|_I, C|_I) = \min_{c \in C}\{\Delta(w|_I, c|_I)\}$ and $\delta(w|_I, C|_I) = \min_{c \in C}\{\delta(w|_I, c|_I)\}$. Given a tester (i.e., a distribution) $\mathcal{D}$ for the code $C \subseteq \mathbb{F}^n$, we let*

$$\rho^{\mathcal{D}}(w) = \mathop{\mathbf{E}}_{I \sim \mathcal{D}}[\delta(w|_I, C|_I)] \quad \textit{be the expected relative local distance of input } w.$$

We say that the tester $\mathcal{D}$ has robustness $\rho^{\mathcal{D}}(C)$ on the code C if for every $w \in \mathbb{F}^n$ it holds that $\rho^{\mathcal{D}}(w) \geq \rho^{\mathcal{D}}(C) \cdot \delta(w, C)$.

Let $\{C_n\}_n$ be a family of codes where C_n has blocklength n and $\mathcal{D}_n$ is a tester for C_n. A family of codes $\{C_n\}_n$ is robustly testable *with respect to testers $\{\mathcal{D}_n\}_n$ if there exists a constant $\alpha > 0$ such that for all n we have $\rho^{\mathcal{D}_n}(C_n) \geq \alpha$.*

We say that a code $C \subseteq \mathbb{F}^n$ is *smooth* with respect to its tester $\mathcal{D}$ if every coordinate of $[n]$ is queried by $\mathcal{D}$ with the same probability, i.e., for all $i, j \in [n]$ it holds that $\mathbf{Pr}_{I \sim \mathcal{D}}[i \in I] = \mathbf{Pr}_{I \sim \mathcal{D}}[j \in I]$.

4.2 Our Corollary - Stable LTCs

In this section we construct stable LTCs, thus proving Theorem 4. In what follows, we shorten notations by omitting the prefix "2-dimensional" before "hyperplane" and "hyperplane tester". For example, when we say "hyperplane tester" we mean 2-dimensional hyperplane tester. We will need the following results of [31] and [34].

Theorem 8 ([31]). *There exists an explicit construction of linear codes $C \subseteq \mathbb{F}_2^n$ such that $\mathrm{rate}(C) = \Omega(1)$, $\delta(C) = \Omega(1)$, and C is encodable in linear time and decodable in linear time from αn errors for some constant $\alpha > 0$.*

Theorem 9 ([34]). *Let $m \geq 3$ be a constant integer and $C \subseteq \mathbb{F}^n$ be a linear code. Let $\mathcal{D}$ be the 2-dimensional hyperplane tester for C^m. Then,*

$$\rho^{\mathcal{D}}(C^m) \geq \frac{(\delta(C))^{3m}}{18^{\log_{1.5} m}}.$$

Moreover, the query complexity of $\mathcal{D}$ is n^2 and C^m is smooth with respect to $\mathcal{D}$, i.e., every coordinate is queried with the same probability. Note that the block-length of C^m is n^m.

Our construction will use a repetitive tester, formulated next.

Definition 8 (Repetitive tester). *Assume that $C \subseteq \mathbb{F}^n$ is efficiently decodable from αn errors and hence by a result of [34] C^2 is efficiently decodable from $\alpha^2 n^2$ errors. Let $\mathcal{D}$ be a 2-dimensional hyperplane tester for C^m. Let $N, \epsilon', \rho' > 0$ such that $\epsilon' \leq \alpha^2$. An (h, ϵ', ρ')-repetitive tester $\mathcal{D}_{rep}$ is a tester that on an input word $M \in \mathbb{F}^{n^m}$:*

- *Invokes $\mathcal{D}$ h times and obtains h hyperplanes $\tau_1, \tau_2, \ldots, \tau_h$.*
- *For every $i \in [h]$, uses a decoder for C^2 to determine (in polynomial time) $\delta(M|_{\tau_i}, C^2)$, where if the decoding fails (i.e., $\delta(M|_{\tau_i}, C^2) > \alpha^2$) then the corresponding estimate is defined as 1.*
- *If the fraction of hyperplanes (among $\tau_1, \tau_2, \ldots, \tau_h$) that are ϵ'-far from C^2 is at least ρ', then $\mathcal{D}_{rep}$ rejects. Otherwise, he accept.*

Remark 2. We note that a repetitive tester uses at most $h \cdot 2m \log(n)$ random bits, since it invokes h times a 2-dimensional hyperplane tester that uses at most $2m \log(n)$ random bits (see Remark 1).

We now state Theorem 10, which implies Theorem 4 as a corollary.

Theorem 10. *Let $m \geq 3$ be a constant integer and $C \subseteq \mathbb{F}_2^n$ be the code from [31] such that* $\mathrm{rate}(C) = \Omega(1)$, $\delta(C) = \Omega(1)$, *and C is efficiently decodable from αn errors, where $\alpha < \delta(C)/2$ is a positive constant. Let $\gamma = \frac{(\delta(C))^{3m}}{18^{\log_{1.5} m}}$. Let δ' and t be any positive numbers satisfying*

- $\delta' \leq \alpha^2$, *and*
- $t \leq \min\left\{\mathrm{rate}(C^m) \cdot 0.0001, \frac{1}{40} \cdot \gamma \cdot \delta'\right\} \cdot n^m = \Omega(n^m)$.

Then, letting $\mathcal{D}_{rep}$ be a (h, ϵ', ρ')-repetitive tester for C^m, where $h = 10^2 20^2$ and $\epsilon' = \frac{1}{4}\gamma\delta'$ and $\rho' = \frac{1}{5}$, we have:

- $\mathrm{rate}(C^m) = (\mathrm{rate}(C))^m = \Omega_m(1)$, $\delta(C^m) = (\delta(C))^m = \Omega_m(1)$, *and the query complexity of $\mathcal{D}_{rep}$ is $O(n^2) = O((\mathrm{blocklength}(C^m))^{\frac{2}{m}})$,*
- *C^m has a t-ZK encoding function E_C, such that* $\mathrm{rate}\,(E_C) = \Omega_m\,(1)$, *and*
- *C^m is $(t, \frac{1}{4}, \delta')$-stable with respect to $\mathcal{D}_{rep}$.*

The proof of Theorem 10 is deferred to the full version.

Acknowledgments. The research of the first, third and fourth authors has received funding from the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC.

The research of the second author was supported in part by a DARPA/ONR PROCEED award, NSF grants 1228984, 1136174, 1118096, 1065276,0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an

equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

1. Agrawal, S.: Verifiable secret sharing in a total of three rounds. Inf. Process. Lett. 112(22), 856–859 (2012)
2. Babai, L., Fortnow, L., Levin, L.A., Szegedy, M.: Checking Computations in Polylogarithmic Time. In: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC), New Orleans, Louisiana, USA, May 5-8, pp. 21–31. ACM (1991)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)
4. Ben-Or, M., Rabin, T.: Verifiable secret sharing and multiparty protocols with honest majority. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC), Seattle, Washigton, USA, May 14-17, pp. 73–85. ACM (1989)
5. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. SIAM Journal on Computing 36(4), 889–974 (2006)
6. Ben-Sasson, E., Harsha, P., Raskhodnikova, S.: Some 3CNF Properties Are Hard to Test. SIAM Journal on Computing 35(1), 1–21 (2005)
7. Ben-Sasson, E., Sudan, M.: Robust locally testable codes and products of codes. Random Struct. Algorithms 28(4), 387–402 (2006)
8. Ben-Sasson, E., Viderman, M.: Composition of Semi-LTCs by Two-Wise Tensor Products. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX and RANDOM 2009. LNCS, vol. 5687, pp. 378–391. Springer, Heidelberg (2009)
9. Ben-Sasson, E., Viderman, M.: Tensor Products of Weakly Smooth Codes are Robust. Theory of Computing 5(1), 239–255 (2009)
10. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19 (1988)
11. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: FOCS, pp. 383–395 (1985)
12. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: STOC, pp. 364–369 (1986)
13. Damgård, I., Ishai, Y.: Scalable secure multiparty computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 501–520. Springer, Heidelberg (2006)
14. Dinur, I.: The PCP theorem by gap amplification. Journal of the ACM 54(3), 12:1–12:44 (2007)
15. Dinur, I., Sudan, M., Wigderson, A.: Robust Local Testability of Tensor Products of LDPC Codes. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX and RANDOM 2006. LNCS, vol. 4110, pp. 304–315. Springer, Heidelberg (2006)

16. Fehr, S., Maurer, U.M.: Linear VSS and distributed commitments based on secret sharing and pairwise checks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 565–580. Springer, Heidelberg (2002)
17. Feldman, J., Malkin, T., Servedio, R.A., Stein, C.: Secure network coding via filtered secret sharing. In: Proceedings of the 42nd Annual Allerton Conference on Communication, Control, and Computing (2004)
18. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: FOCS, pp. 427–437. IEEE Computer Society (1987)
19. Fitzi, M., Garay, J.A., Gollakota, S., Pandu Rangan, C., Srinathan, K.: Round-optimal and efficient verifiable secret sharing. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 329–342. Springer, Heidelberg (2006)
20. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: STOC, pp. 580–589 (2001)
21. Goldreich, O.: Short Locally Testable Codes and Proofs (Survey). Electronic Colloquium on Computational Complexity (ECCC) (014) (2005)
22. Goldreich, O.: Three XOR-lemmas — an exposition. In: Goldreich, O. (ed.) Studies in Complexity and Cryptography. LNCS, vol. 6650, pp. 248–272. Springer, Heidelberg (2011)
23. Goldreich, O., Sudan, M.: Locally testable codes and PCPs of almost-linear length. Journal of the ACM 53(4), 558–655 (2006)
24. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Extracting correlations. In: FOCS, pp. 261–270 (2009)
25. Ishai, Y., Mahmoody, M., Sahai, A.: On efficient zero-knowledge PCPs. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 151–168. Springer, Heidelberg (2012)
26. Kilian, J., Petrank, E., Tardos, G.: Probabilistically checkable proofs with zero knowledge. In: STOC, pp. 496–505 (1997)
27. Kumaresan, R., Patra, A., Pandu Rangan, C.: The round complexity of verifiable secret sharing: The statistical case. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 431–447. Springer, Heidelberg (2010)
28. Meir, O.: Combinatorial Construction of Locally Testable Codes. SIAM J. Comput. 39(2), 491–544 (2009)
29. Patra, A., Choudhary, A., Rabin, T., Pandu Rangan, C.: The round complexity of verifiable secret sharing revisited. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 487–504. Springer, Heidelberg (2009)
30. Peng, K.: Efficient VSS free of computational assumption. J. Parallel Distrib. Comput. 71(12), 1592–1597 (2011)
31. Spielman, D.A.: Linear-time encodable and decodable error-correcting codes. IEEE Transactions on Information Theory 42(6), 1723–1731 (1996)
32. Trevisan, L.: Some Applications of Coding Theory in Computational Complexity (September 23, 2004)
33. Vazirani, U.V.: Randomness, adversaries and computation. Ph.D. Thesis, EECS, UC Berkeley
34. Viderman, M.: A combination of testability and decodability by tensor products. In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) APPROX/RANDOM 2012. LNCS, vol. 7408, pp. 651–662. Springer, Heidelberg (2012)

A Tight Lower Bound for High Frequency Moment Estimation with Small Error

Yi Li[1] and David P. Woodruff[2]

[1] Department of EECS, University of Michigan, Ann Arbor
leeyi@umich.edu
[2] IBM Almaden Research Center
dpwoodru@us.ibm.com

Abstract. We show an $\Omega((n^{1-2/p}\log M)/\epsilon^2)$ bits of space lower bound for $(1+\epsilon)$-approximating the p-th frequency moment $F_p = \|x\|_p^p = \sum_{i=1}^n |x_i|^p$ of a vector $x \in \{-M, -M+1, \ldots, M\}^n$ with constant probability in the turnstile model for data streams, for any $p > 2$ and $\epsilon \geq 1/n^{1/p}$ (we require $\epsilon \geq 1/n^{1/p}$ since there is a trivial $O(n \log M)$ upper bound). This lower bound matches the space complexity of an upper bound of Ganguly for any $\epsilon < 1/\log^{O(1)} n$, and is the first of any bound in the long sequence of work on estimating F_p to be shown to be optimal up to a constant factor for any setting of parameters. Moreover, our technique improves the dependence on ϵ in known lower bounds for cascaded moments, also known as mixed norms. We also continue the study of tight bounds on the dimension of linear sketches (drawn from some distribution) required for estimating F_p over the reals. We show a dimension lower bound of $\Omega(n^{1-2/p}/\epsilon^2)$ for sketches providing a $(1+\epsilon)$-approximation to $\|x\|_p^p$ with constant probability, for any $p > 2$ and $\epsilon \geq 1/n^{1/p}$. This is again optimal for $\epsilon < 1/\log^{O(1)} n$.

1 Introduction

In the standard turnstile model of data streams [1, 2], there is an underlying n-dimensional vector x, which we sometimes refer to as the frequency vector, which is initialized to the zero vector and which evolves through a sequence of additive updates to its coordinates. These updates are fed into a streaming algorithm, and have the form $x_i \leftarrow x_i + \delta$, changing the i-th coordinate by the value δ. Here δ is an arbitrary positive or negative integer, and x is guaranteed to satisfy the promise that at all times $x \in \{-M, -M+1, \ldots, M\}^n$. The goal of the streaming algorithm is to make a small number of passes over the data and to use limited memory to compute statistics of x, such as the frequency moments [3], the number of distinct elements [4], the empirical entropy [5], and the heavy hitters [6, 7]. Since computing these statistics exactly or deterministically requires a prohibitive $\Omega(n)$ bits of space [3], these algorithms are both randomized and approximate. For most of these problems in the turnstile model, they are quite often studied in the model in which the data stream algorithm can only make a single pass over the data. This is critical in many online applications, such as

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 623–638, 2013.

Table 1. Results are in terms of bits and for constant $p > 2$. Here, $g(p,n) = \min_{c \text{ constant}} g_c(n)$, where $g_1(n) = \log n$, $g_c(n) = \log(g_{c-1}(n))/(1-2/p)$. For brevity, we only list those results which work in the general turnstile model, and for which bounds for general ϵ have been derived. For other recent interesting work, we refer the reader to [20], which requires the insertion-only model and does not have bounds for general $\epsilon > 0$. We also start the upper bound timeline with [8], since that is the first work which achieved an exponent of $1 - 2/p$ for n. For earlier works which achieved worse exponents for n, see [3, 21–23]. We note that [3] initiated the problem and obtained an $O(n^{1-1/p}\epsilon^{-2}\log(M))$ bound in the insertion-only model. We also omit from the table previous lower bounds which hold for linear sketches rather than for the turnstile model [24, 25], though these are discussed in the Introduction.

F_p Algorithm	Space Complexity
[8]	$O(n^{1-2/p}\epsilon^{-O(1)}\log^{O(1)} n \log(M))$
[9]	$O(n^{1-2/p}\epsilon^{-2-4/p}\log n \log^2(M))$
[10]	$O(n^{1-2/p}\epsilon^{-O(1)}\log^{O(1)} n \log(M))$
[11]	$O(n^{1-2/p}\epsilon^{-2-6/p}\log n \log(M))$
[12]	$O(n^{1-2/p}\epsilon^{-2-4/p}\log n \cdot g(p,n)\log(M))$
[13]	$O(n^{1-2/p}\log n \log(M)\epsilon^{-O(1)})$
[14], **Best upper bound**	$O(n^{1-2/p}\epsilon^{-2}\log n \cdot \log(M)/\min(\log n, \epsilon^{4/p-2})))$
[3]	$\Omega(n^{1-5/p})$
[15]	$\Omega(\epsilon^{-2})$
[16]	$\Omega(n^{1-2/p-\gamma}\epsilon^{-2/p})$, any constant $\gamma > 0$
[17]	$\Omega(n^{1-2/p}\epsilon^{-2/p})$
[18]	$\Omega(n^{1-2/p}\epsilon^{-4/p}/\log^{O(1)} n)$
[19]	$\Omega(n^{1-2/p}\epsilon^{-2}/\log n)$
This paper	$\Omega(n^{1-2/p}\epsilon^{-2}\log(M))$

network traffic monitoring, and when most of the data resides on an external disk, for which multiple passes over it is too costly. In this paper we focus on one-pass streaming algorithms.

We show new lower bounds for approximating the p-th frequency moment F_p, $p > 2$, in a data stream. In this problem the goal is to estimate $\sum_{i=1}^{n} |x_i|^p$ up to a factor of $1 + \epsilon$ with constant probability, where $x \in \{-M, -M+1, \ldots, M\}^n$ and we make the standard assumption that $\log(Mn) = \Theta(\log M)$ and $p > 2$ is a constant. We summarize the sequence of work on this problem in Table 1.

The previous best upper bound is $O(n^{1-2/p}\epsilon^{-2}\log n \log M/\min(\log n, \epsilon^{4/p-2}))$, due to Ganguly [14]. Notice that for $\epsilon < 1/\log^{O(1)} n$, this bound simplifies to $O(n^{1-2/p}\epsilon^{-2}\log M)$. The previous best lower bound is due to [17, 19], and is $\Omega(n^{1-2/p}\epsilon^{-2}/\log n + n^{1-2/p}\epsilon^{-2/p})$. We improve the space complexity lower bound, in bits, to $\Omega(n^{1-2/p}\epsilon^{-2}\log M)$ for any $\epsilon > 1/n^{1/p}$ (we require $\epsilon > 1/n^{1/p}$ since there is a trivial $O(n \log M)$ upper bound). In light of the upper bound given above, our lower bound is optimal for any $\epsilon < 1/\log^{O(1)} n$ and constant $p > 2$. This is an important range of parameters; even in applications with 1% error, i.e., $\epsilon = .01$, we have that for, e.g., $n = 2^{32}$, $\epsilon < 1/\log n$. Understanding the limitations of streaming algorithms in terms of ϵ is also the focus of a body of

work in the streaming literature, see, for example, [15, 26–34]. Our lower bound gives the first asymptotically optimal bound for any setting of parameters in the long line of work on estimating F_p in a data stream.

A few recent works [24, 25] also study the "sketching model" of F_p-estimation in which the underlying vector x is in $\mathbb{R}^n$, rather than in the discrete set $\{-M, -M+1, \ldots, M\}^n$. One seeks a distribution over linear maps $A : \mathbb{R}^n \to \mathbb{R}^s$, for some $s \ll n$, so that for any fixed vector $x \in \mathbb{R}^n$, one can $(1+\epsilon)$-approximate $\|x\|_p^p$ with constant probability by applying an estimation procedure $E : \mathbb{R}^s \to \mathbb{R}$ to Ax. One seeks the smallest possible s for a given ϵ and n. Lower bounds in the turnstile model do not imply lower bounds in the sketching model. Indeed, if the input vector $x \in \{-M, -M+1, \ldots, M\}^n$, then the inner product of x with the single vector $(1, 1/(M+1), 1/(M+1)^2, \ldots, 1/(M+1)^{n-1})$ is enough to recover x, so a sketching dimension of $s = 1$ suffices. Previously, it was known that $s = \Omega(n^{1-2/p})$ [25], which for constant $p > 2$ and constant $\epsilon > 0$ was recently improved to $s = \Omega(n^{1-2/p} \log n)$ [24]. We note that the upper bound of [14] is a linear sketch with $s = O(n^{1-2/p}\epsilon^{-2})$ dimensions for any $\epsilon < 1/\log^{O(1)} n$. We improve the lower bound on s for general ϵ, obtaining an $s = \Omega(n^{1-2/p}\epsilon^{-2})$ lower bound. Our lower bound matches the upper bound of [14] for $\epsilon < 1/\log^{O(1)} n$ up to a constant factor, and improves the lower bound of [24] for $\epsilon < 1/\log^{O(1)} n$.

Our Approach: To prove our lower bound in the turnstile model, we define a variant of the ℓ_∞^k communication problem [16]. In this problem there are two parties, Alice and Bob, holding vectors $x, y \in \{-M, -M+1, \ldots, M\}^n$ respectively, and their goal is to decide if $\|x - y\|_\infty = \max_{i \in [n]} |(x-y)_i| \leq 1$ or there exists a unique $i \in [n]$ for which $|(x-y)_i| \geq k$ and for all $j \neq i$, $|(x-y)_j| \leq 1$. The standard reduction to frequency moments is to set $k = \epsilon^{1/p} n^{1/p}$, from which one can show that any streaming algorithm for outputting a $(1+\epsilon)$-approximation to F_p can be used to build a communication protocol for solving ℓ_∞^k with communication proportional to the algorithm's space complexity. Using the communication lower bound of $\Omega(n/k^2)$ for the ℓ_∞^k problem, this gives the bound $\Omega(n^{1-2/p}\epsilon^{-2/p})$.

Our first modification is to instead set $k = \epsilon n^{1/p}$, which gives a communication lower bound of $\Omega(n^{1-2/p}\epsilon^{-2})$. However, the reduction from approximating F_p no longer works. To remedy this, we introduce a third player Charlie whose input is $z \in \{0^n, n^{1/p}\mathbf{e}_1, \ldots, n^{1/p}\mathbf{e}_n\}$, where $\mathbf{e}_i$ denotes the i-th standard unit vector, and we seek a $(1+\epsilon)$-approximation to $\|x - y + z\|_\infty$. The main point is that if $|x_i - y_i| = \epsilon n^{1/p}$, then $\|x - y + z\|_\infty$ differs by a factor of $1 + \epsilon$ depending on whether or not Charlie's input is $n^{1/p}\mathbf{e}_i$, 0^n, or $n^{1/p}\mathbf{e}_j$ for some $j \neq i$. Note that Charlie has no information as to whether $|x_i - y_i| = k$ or $|x_i - y_i| \leq 1$, which is determined by Alice and Bob's inputs. One can think of this as an extension to the classical indexing problem, which involves two players, in which the first player has a string $x \in \{0,1\}^n$, the second player an index $i \in [n]$, and the second player needs to output x_i. Now, we have Alice and Bob solving multiple single-coordinate problems and Charlie is indexing into one of these problems.

This modification allows us to strengthen the problem for use in applications. We choose the legal inputs x, y, z to the three-player problem to have the following promise: (1) $\|x - y + z\|_\infty \leq 1$, (2) there is a unique i for which $|(x-y+z)_i| = \epsilon n^{1/p}$ and all other $j \neq i$ satisfy $|(x-y+z)_j| \leq 1$, or (3) there is a unique i for which $|(x-y+z)_i|$ is either $(1+\epsilon)n^{1/p}$ or $(1-\epsilon)n^{1/p}$ and all other $j \neq i$ satisfy $|(x-y+z)_j| \leq 1$. Using the 1-way property of a communication protocol, we can adapt the argument in [16] for the ℓ_∞^k problem to show an $\Omega(n^{1-2/p}\epsilon^{-2})$ lower bound for this 3-player problem. Here we use the intuitive fact that Alice and Bob need to solve the ℓ_∞^k problem with $k = \epsilon n^{1/p}$ because if Charlie has the input $z = n^{1/p}\mathbf{e}_i$, then $\|x-y+z\|_\infty$ differs by a factor of $(1+\epsilon)$ depending on whether $|(x-y)_i| = \epsilon n^{1/p}$ or $|(x-y)_i| \leq 1$. Moreover, Alice and Bob have no information about z since the protocol is 1-way.

We show that a streaming algorithm providing a $(1+\epsilon)$-approximation to F_p can decide which of the three cases the input is in by invoking it twice in the reduction to the communication problem. Here, Alice, Bob, and Charlie create local streams σ_A, σ_B, and σ_C, Alice sends the state of the algorithm on σ_A to Bob, who computes the state of the algorithm on $\sigma_A \circ \sigma_B$ who sends it to Charlie. Charlie then queries a $(1+\epsilon)$-approximate F_p value of $\sigma_A \circ \sigma_B$, together with a $(1+\epsilon)$-approximate F_p value of $\sigma_A \circ \sigma_B \circ \sigma_C$. Assuming both query responses are correct, we can solve this new communication problem, yielding an $\Omega(n^{1-2/p}\epsilon^{-2})$ bits of space lower bound.

To improve the space further, we define an augmented version of this 3-player problem, in which Alice, Bob, and Charlie have $r = \Theta(\log M)$ independent instances of this problem, denoted x^i, y^i, z^i, for $i \in [r]$. Charlie additionally has an index $I \in [r]$ together with (x^i, y^i) for all $i > I$. His goal is to solve the I-th instance of the communication problem. This problem can be seen as an extension to the classical augmented indexing problem, which involves two players, in which the first player has a string $x \in \{0,1\}^n$, the second player an index $i \in [n]$ together with $x_{i+1}, \ldots, x_n$, and the second player needs to output x_i. We now have a "functional" version of augmented indexing, in which Alice and Bob solve multiple instances of a problem, and Charlie's input indexes one of these problems. Via a direct sum argument [16, 35], we show our problem has randomized communication complexity $\Omega(n^{1-2/p}\epsilon^{-2}\log M)$. Finally, we show how a streaming algorithm for $(1+\epsilon)$-approximating F_p can be used to solve this augmented problem.

We believe our technique will improve the dependence on ϵ in space lower bounds for other problems in the data stream literature. For example, we can improve the dependence on ϵ in known lower bounds for estimating cascaded moments, also known as mixed norms [11, 36, 37]. Here there is an underlying $n \times d$ matrix A, and the goal is to estimate $\ell_p(\ell_q)(A) = (\sum_{i=1}^n \|A_i\|_q^p)^{1/p}$, where A_i is the i-th row of A. In [37] (beginning of Section 2) a lower bound of $\Omega(n^{1-2/p}d^{1-2/q})$ is shown for constant ϵ and $p, q \geq 2$ via a reduction to the so-called t-player set disjointness problem for $t = 2n^{1/p}d^{1/q}$. Straightforwardly setting $t = \Theta(\epsilon^{1/k}n^{1/p}d^{1/q})$, their proof establishes a lower bound of $\Omega(n^{1-2/p}d^{1-2/q}\epsilon^{-2/p})$ for general ϵ. Our technique also applies to the t-player

set disjointness problem, by introducing a $(t+1)$-st player Charlie with an input $z \in \{0^{nd}, n^{1/p}d^{1/q}\mathbf{e}_i\mathbf{e}_j^T$ for $i \in [n], j \in [d]\}$, and applying analogous ideas to those given above. This results in a new lower bound of $\Omega(n^{1-2/p}d^{1-2/q}\epsilon^{-2})$. The same ideas apply to improving the $\Omega(n^{1/2})$ lower bound for $\ell_2(\ell_0)(A)$ given in [37] (here $\|A_i\|_0$ denotes the number of non-zero entries of A_i). A straightforward adaptation of the arguments in [37] for general ϵ gives a lower bound of $\Omega(n^{1/2}\epsilon^{-1/2})$, while our technique strengthens this to $\Omega(n^{1/2}\epsilon^{-1})$. We sketch these improvements in the full version of this paper.

Our lower bound in the sketching model is simpler and perhaps surprising. We consider two cases: the input $x \in \mathbb{R}^n$ is equal to $g + n^{1/p}\mathbf{e}_i$ for a vector g of i.i.d. standard normal random variables and a random standard unit vector $\mathbf{e}_i$, or the input x is equal to $g' + n^{1/p}(1+\epsilon)\mathbf{e}_i$ for a vector g' of i.i.d. standard normal random variables. By Yao's minimax principle, there exists a fixed $s \times n$ sketching matrix A for which the variation distance between distributions $A(g+n^{1/p}\mathbf{e}_i)$ and $A(g' + n^{1/p}(1+\epsilon)\mathbf{e}_i)$ is large. Since we can, without loss of generality, assume the rows of A are orthonormal (given Ax, one can always compute LAx for any change of basis matrix L for the rowspace of A), this implies the variation distance between $h + n^{1/p}A_i$ and $h' + n^{1/p}(1+\epsilon)A_i$ is large, where h, h' are s-dimensional vectors of i.i.d. standard normal random variables and A_i is the i-th column of A. However, for a random i, since the rows of A are orthonormal, $\|A_i\|_2$ is only about $O(\sqrt{s/n})$. For such i, this contradicts a standard variation distance upper bound between two shifted s-dimensional Gaussian vectors unless $s = \Omega(n^{1-2/p}/\epsilon^2)$.

2 Preliminaries

Notations. We denote the canonical basis of $\mathbb{R}^n$ by $\{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$. Let $[n]$ denote the set $\{1, \ldots, n\}$. For a vector $v \in \mathbb{R}^n$ and an index set $K \subset [n]$, define a vector in $\mathbb{R}^n$, denoted by $v|_K$, such that $(v|_K)_i = v_i$ for all $i \in K$ and $(v|_K)_i = 0$ for all $i \notin K$.

Probability. For a random variable X and a probability distribution $\mathcal{D}$, we write $X \sim \mathcal{D}$ for X being subject to the distribution $\mathcal{D}$. We denote the multivariate Gaussian with mean μ and covariance matrix Σ by $N(\mu, \Sigma)$. Let I_n denote the identity matrix of size $n \times n$.

We shall need the following lemma regarding concentration of Gaussian measure, see Chapter 1 of [38].

Lemma 1. *Suppose that $X \sim N(0, I_n)$ and the function $f : \mathbb{R}^n \to \mathbb{R}$ is 1-Lipschitz, i.e., $|f(x) - f(y)| \leq \|x - y\|_2$ for all $x, y \in \mathbb{R}^n$. Then for any $t > 0$ it holds that $\Pr_x\{|f(x) - \mathbb{E}f(x)| > t\} \leq 2e^{-t^2/2}$.*

Definition 1. *Suppose μ and ν are two probability measures over some Borel algebra $\mathcal{B}$ on $\mathbb{R}^n$. Then the total variation distance between μ and ν is defined as*

$$d_{TV}(\mu, \nu) = \sup_{B \in \mathcal{B}} |\mu(B) - \nu(B)| \left(= \frac{1}{2}\int_x |f(x) - g(x)|dx \right),$$

where the second equality holds when μ and ν have probability density functions $f(x)$ and $g(x)$ respectively.

The following is a result ([39]) that bounds the total variation distance between two multivariate Gaussian distributions.

Proposition 1. $d_{TV}(N(\mu_1, I_n), N(\mu_2, I_n)) \leq \|\mu_1 - \mu_2\|_2/\sqrt{2}$.

Communication Model. We briefly summarize the notions from communication complexity that we will need. For more background on communication complexity, we refer the reader to [40]. In this paper we consider a one-way communication model. There are three players Alice, Bob and Charlie with private random coins. Alice is given an input x, Bob y and Charlie z, and their goal is to compute a function $f(x, y, z)$. Alice sends exactly one message to Bob and Bob sends exactly one message to Charlie, according to a protocol Π, and then Charlie outputs an answer. We say the protocol Π is δ-error if for every legal triple (x, y, z) of inputs, the answer equals $f(x, y, z)$ with probability at least $1 - \delta$, where the probability is taken over the random coins of the players. The concatenation of the message sent from Alice to Bob with the message from Bob to Charlie, as well as Charlie's output, is called the *transcript* of Π. The maximum length of the transcript (in bits) is called the *communication cost* of Π. The *communication complexity* of f is the minimal communication cost of a δ-error protocol for f, and is denoted $R_\delta(f)$.

Mutual Information. Let (X, Y) be a pair of discrete random variables with joint distribution $p(x, y)$. The mutual information $I(X; Y)$ is defined as $I(X; Y) = \sum_{x,y} p(x, y) \log \frac{p(x,y)}{p(x)p(y)}$, where $p(x)$ and $p(y)$ are marginal distributions. The following are basic properties regarding mutual information.

Proposition 2. *Let X, Y, Z be discrete random variables defined on $\Omega_X, \Omega_Y, \Omega_Z$, respectively, and let f be a function defined on Ω. Then*

1. *$I(X; Y) \geq 0$ and the equality is attained iff X and Y are independent;*
2. *Chain rule for mutual information: $I(X, Y; Z) = I(X; Z) + I(X; Y|Z)$;*
3. *Data processing inequality: $I(f(X); Y) \leq I(X; Y)$.*

2.1 Direct-sum Technique

The following definitions and results are from [16]. See also Section 6 of [41].

Definition 2. *Let Π be a randomized protocol with inputs belonging to a set $\mathcal{K}$. We shall abuse notation and also use $\Pi(X, Y, Z)$ to denote the transcript of protocol Π, which is a random variable which also depends on the private coins of the players. When X, Y, Z are understood from context, we sometimes further abbreviate $\Pi(X, Y, Z)$ as Π. Let μ be a distribution on $\mathcal{K}$ and suppose that $(X, Y, Z) \sim \mu$. The* information cost *of Π with respect to μ is defined to be $I(X, Y, Z; \Pi(X, Y, Z))$.*

Definition 3. *The δ-error information complexity of f with respect to a distribution μ, denoted by $IC_{\mu,\delta}(f)$, is defined to be the minimum information cost of a δ-error protocol for f with respect to μ.*

Using this definition, it follows immediately that (see [16]):

Proposition 3. *$R_\delta(f) \geq IC_{\mu,\delta}(f)$ for any distribution μ and $\delta > 0$.*

Definition 4 (Conditional information cost). *Let Π be a randomized protocol whose inputs belong to some set $\mathcal{K}$ of valid inputs and that ζ is a mixture of product distributions on $\mathcal{K} \times \mathcal{W}$. Suppose that $((X, Y, Z), W) \sim \zeta$. The conditional information cost of Π with respect to ζ is defined as $I(X, Y, Z; \Pi(X, Y, Z)|W)$.*

Definition 5 (Conditional information complexity). *The δ-error conditional information complexity of f with respect to ζ, denoted $CIC_{\zeta,\delta}(f)$, is defined to be the minimum conditional information cost of a δ-error protocol for f with respect to ζ.*

Definition 6 (Decomposable functions). *Suppose that f is a function defined on $\mathcal{L}^n$. We say that f is g-decomposable with primitive h if it can be written as $f(\mathbf{x}, \mathbf{y}, \mathbf{z}) = g(h(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1), \ldots, h(\mathbf{x}_n, \mathbf{y}_n, \mathbf{z}_n))$ for some function h defined on $\mathcal{L} \to \mathcal{Q}$ and g on $\mathcal{Q}^n$. Sometimes we simply say that f is decomposable with primitive h.*

Definition 7 (Embedding). *For a vector $w \in \mathcal{L}^n$, $j \in [n]$ and $u \in \mathcal{L}$, we define $embed(w, j, u)$ to be the n-dimensional vector over $\mathcal{L}$ whose i-th component is defined as follows: $embed(w, j, u)_i = w_i$ if $i \neq j$; and $embed(w, j, u)_i = u$ if $i = j$.*

Definition 8 (Collapsing distribution). *Suppose f is g-decomposable with primitive h. We call $(x, y, z) \in \mathcal{L}^n$ a collapsing input for f, if for every j and $(u, v, w) \in \mathcal{L}$, it holds that*

$$f(embed(x, j, u), embed(y, j, v), embed(z, j, w)) = h(u, v, w).$$

We call a distribution μ on $\mathcal{L}^n$ collapsing for f if every (x, y, z) in the support of μ is a collapsing input.

Lemma 2 (Information cost decomposition). *Let Π be a protocol whose inputs belong to $\mathcal{L}^n$ for some set $\mathcal{L}$. Let ζ be a mixture of product distributions on $\mathcal{L} \times \mathcal{D}$ and suppose that $((X, Y, Z), D) \sim \zeta^n$. Then, $I(X, Y, Z; \Pi(X, Y, Z)|D) \geq \sum_i I(X_i, Y_i, Z_i; \Pi(X, Y, Z)|D)$.*

Lemma 3 (Reduction lemma). *Let Π be a δ-error protocol for a decomposable function f defined on $\mathcal{L}^n$ with primitive h. Let ζ be a mixture of product distributions on $\mathcal{L} \times \mathcal{D}$, let $\eta = \zeta^n$, and suppose that $((X, Y, Z), D) \sim \eta$. If the distribution of (X, Y, Z) is a collapsing distribution for f, then for all $j \in [n]$, it holds that $I((X_j, Y_j, Z_j); \Pi(X, Y, Z)|D) \geq CIC_{\zeta,\delta}(h)$.*

2.2 Hellinger Distance

Definition 9. *The Hellinger distance $h(P,Q)$ between probability distributions P and Q on a domain Ω is defined by*

$$h^2(P,Q) = 1 - \sum_{\omega\in\Omega} \sqrt{P(\omega)Q(\omega)} = \frac{1}{2}\sum_{\omega\in\Omega}(\sqrt{P(\omega)} - \sqrt{Q(\omega)})^2.$$

One can verify that the Hellinger distance is a metric satisfying the triangle inequality, see, e.g., [16]. The following proposition connects the Hellinger distance and the total variation distance.

Proposition 4. *(see, e.g., [41])* $h^2(P,Q) \le d_{TV}(P,Q) \le \sqrt{2}h(P,Q)$.

In connection with mutual information, we have that

Lemma 4 ([16]). *Let F_{z_1} and F_{z_2} be two random variables. Let Z denote a random variable with uniform distribution in $\{z_1, z_2\}$ Suppose $F(z)$ is independent of Z for each $z \in \{z_1, z_2\}$. Then, $I(Z; F(Z)) \ge h^2(F_{z_1}, F_{z_2})$.*

In [41], it is shown that a randomized private-coin three-party protocol exhibits the rectangle property in the following sense: there exist functions q_1, q_2, q_3 such that for all legal inputs (x, y, z) and transcripts τ, it holds that

$$\Pi_{x,y,z}(\tau) = q_1(x,\tau)q_2(y,\tau)q_3(z,\tau).$$

The following is a variant of the inverse triangle inequality in [16] that we need to accommodate our setting of three players. The proof is similar to that for two players and thus we omit it because of space limitations.

Lemma 5 (Inverse triangle inequality). *For any randomized protocol Π and for any inputs x, y, z and x', y', z it holds that*

$$h^2(\Pi_{x,y,z}, \Pi_{x',y,z}) + h^2(\Pi_{x,y',z}, \Pi_{x',y',z}) \le 2h^2(\Pi_{x,y,z}, \Pi_{x',y',z}).$$

3 Augmented L_∞ Promise Problem

In this section we define the Augmented L_∞ Promise Problem. First, though, we consider a slightly different gap problem than that considered in [16] for a problem which we refer to as the L_∞ Promise problem.

Definition 10 ($L_\infty(k,\epsilon)$). *Assume that $\epsilon k \ge 1$. There are three players Alice, Bob and Charlie in the one-way communication model with private coins. Alice receives a vector $\mathbf{a} \in \{0,\ldots,\epsilon k\}^n$, Bob a vector $\mathbf{b} \in \{0,\ldots,\epsilon k\}^n$ and Charlie both an index $j \in [n]$ and a bit $c \in \{0,1\}$. The input is guaranteed to satisfy $|\mathbf{a}_i - \mathbf{b}_i| \le 1$ for all $j \ne i$. Charlie is asked to decide which three of the following cases happen, provided we are promised that the input is indeed in one of these three cases: (1) $(\mathbf{a}-\mathbf{b})_j + ck \le 1$; (2) $(\mathbf{a}-\mathbf{b})_j + ck = (1-\epsilon)k$; (3) $(\mathbf{a}-\mathbf{b})_j + ck \ge k$. Charlie's output must be correct with probability $\ge 9/10$.*

In the definition above, the index j is referred to as the *spike* position.

We consider the following distribution μ on the input. Let $c = 0$. Define the random variable $((X, Y), D)$ as follows. The random variable is uniform on $\{0, \ldots, k\} \times \{0, 1\} \setminus \{(0, 1), (k, 0)\}$. If $D = (d, 0)$ then $X = d$ and Y is uniform on $\{d, d+1\}$; if $D = (d, 1)$ then $Y = d$ and X is uniformly distributed on $\{d-1, d\}$.

Theorem 1. $R(L_\infty(k, \epsilon)) = \Omega(n/(k^2\epsilon^2))$.

Proof. Making $c = 0$ in Charlie's input, we see that μ^n is a collapsing distribution for $L_\infty(k, \epsilon)$ so we can apply the direct sum technique. Letting $\mathbf{x}_i = (\mathbf{a}_i, \mathbf{b}_i)$, it follows that

$$R(L_\infty(k, \epsilon)) \geq \sum_{i=1}^{n} I(\mathbf{x}_1, \ldots, \mathbf{x}_n; \Pi(\mathbf{x}_1, \ldots, \mathbf{x}_n) | D_1, \ldots, D_n) \geq nCIC_\mu(L^1_\infty(k, \epsilon)),$$

where $L^1_\infty(k, \epsilon)$ is the single coordinate problem of $L_\infty(k, \epsilon)$, that is, the $L_\infty(k, \epsilon)$ problem with $n = 1$. Therefore, it suffices to show that

$$CIC_\mu(L^1_\infty(k, \epsilon)) = \Omega\left(1/(k^2\epsilon^2)\right). \tag{1}$$

This is a single-coordinate problem, and we shall drop the index i henceforth in the proof. Let U_d denote a random variable with uniform distribution on $\{d, d+1\}$.

$$\begin{aligned}
&CIC_\mu(L^1_\infty(k, \epsilon)) \\
&= I(\mathbf{x}; \Pi(\mathbf{x}) | D) \\
&= \frac{1}{2\epsilon k}\left(\sum_{d=0}^{\epsilon k-1} I(U_d; \Pi(d, U_d)) + \sum_{d=1}^{\epsilon k} I(U_{d-1}; \Pi(U_{d-1}, d))\right) \\
&\geq \frac{1}{2\epsilon k}\left(\sum_{d=0}^{\epsilon k-1} h^2(\Pi_{d,d,0}, \Pi_{d,d+1,0}) + \sum_{d=0}^{\epsilon k-1} h^2(\Pi_{d-1,d,0}, \Pi_{d,d,0})\right) &(2)\\
&\geq \frac{1}{4\epsilon^2 k^2}\left(\sum_{d=0}^{\epsilon k-1} h(\Pi_{d,d,0}, \Pi_{d,d+1,0}) + \sum_{d=0}^{\epsilon k-1} h(\Pi_{d-1,d,0}, \Pi_{d,d,0})\right)^2 &(3)\\
&\geq \frac{1}{4\epsilon^2 k^2} h^2(\Pi_{0,0,0}, \Pi_{\epsilon k,\epsilon k,0}) &(4)
\end{aligned}$$

where we used Lemma 4 for (2), the Cauchy-Schwarz inequality for (3) and the triangle inequality for (4), By the three-player version of the inverse triangle inequality (Lemma 5),

$$\begin{aligned}
h^2(\Pi_{0,0,0}, \Pi_{\epsilon k,\epsilon k,0}) &\geq \frac{1}{2}(h^2(\Pi_{0,0,0}, \Pi_{\epsilon k,0,0}) + h^2(\Pi_{0,\epsilon k,0}, \Pi_{\epsilon k,\epsilon k,0})) \\
&\geq \frac{1}{2}h^2(\Pi_{0,\epsilon k,0}, \Pi_{\epsilon k,\epsilon k,0}) \geq \frac{1}{4}d^2_{TV}(\Pi_{0,\epsilon k,0}, \Pi_{\epsilon k,\epsilon k,0}), \qquad (5)
\end{aligned}$$

where we used Proposition 4 for the last inequality. We now claim that

$$d_{TV}(\Pi_{0,\epsilon k,0}, \Pi_{\epsilon k,\epsilon k,0}) = \Omega(1). \tag{6}$$

Consider the message sent from Alice to Bob, together with the message sent from Bob to Charlie. Let us denote the concatenation of these two messages by $T = T(x, y)$. Notice that the messages do not depend on Charlie's input. We in fact claim a stronger statement than (6), namely that $d_{TV}(T(0, \epsilon k), T(\epsilon k, \epsilon k)) = \Omega(1)$.

To see this, suppose that Charlie's input bit equals 1. Then he needs to decide if the players inputs are in case (2) or in case (3). Let $\mathcal{T}$ be the set of messages from Alice and Bob and from Bob to Charlie that make Charlie output "case (2)" with probability $\geq 3/4$, over his private coins. Then by the correctness of the protocol, $\Pr\{T(0, \epsilon k) \in \mathcal{T}\} \geq \frac{3}{5}$ and $\Pr\{T(\epsilon k, \epsilon k) \in \mathcal{T}\} \leq \frac{2}{15}$. Indeed, otherwise if $\Pr\{T(0, \epsilon k) \in \mathcal{T}\} < 3/5$ then Charlie outputs "case (2)" with probability $< 3/5 + 3/4 \cdot 2/5 = 9/10$, contradicting the correctness of the protocol, while if $\Pr\{T(\epsilon k, \epsilon k) \in \mathcal{T}\} > 2/15$ then Charlie outputs "case (2)" with probability $> 2/15 \cdot 3/4 = 1/10$, again contradicting the correctness of the protocol. Therefore

$$\begin{aligned} d_{TV}(T(0, \epsilon k), T(\epsilon k, \epsilon k)) &\geq |\Pr(T(0, \epsilon k) \in \mathcal{T}) - \Pr(T(\epsilon k, \epsilon k) \in \mathcal{T})| \\ &\geq 3/5 - 2/15 = \Omega(1), \end{aligned}$$

whence (6) follows since $d_{TV}(\Pi_{0,\epsilon k,0}, \Pi_{\epsilon k,\epsilon k,0}) \geq d_{TV}(T(0, \epsilon k), T(\epsilon k, \epsilon k))$.

Plugging (6) into (5) and then (5) into (4), we have that (1) follows immediately. □

Now we define a stronger problem called the Augmented L_∞ Promise problem, and denoted by Aug-$L_\infty(r, k, \epsilon)$. We further abbreviate this by Aug-$L_\infty(r, k)$ when ϵ is clear from the context.

Definition 11 (Aug-$L_\infty(r, k, \epsilon)$). *Consider r instances of $L_\infty(k, \epsilon)$, denoted $(\mathbf{a}_1, \mathbf{b}_1, j_1, c_1)$, ..., $(\mathbf{a}_r, \mathbf{b}_r, j_r, c_r)$. In addition to these inputs, Charlie has an index $I \in [r]$, together with $\mathbf{a}_j$ and $\mathbf{b}_j$ for all $j > I$. The goal is to decide for the I-th $L_\infty(k)$ instance, which of the three cases the input is in, with probability $\geq 5/8$. The input is guaranteed to satisfy $c_i = 0$ for all $i \neq I$.*

Now we define a distribution ν on the inputs to the Aug-$L_\infty(r, k)$ problem: the r instances of $L_\infty(k)$ are independent hard instances (i.e., drawn from μ) of $L_\infty(k)$. The index I is uniformly random on the set $[r]$.

Theorem 2. *$R(L_\infty(r, k, \epsilon)) = \Omega(nr/(k^2\epsilon^2))$.*

Proof. Write $\mathbf{x}_i = (\mathbf{a}_i, \mathbf{b}_i)$. It suffices to show that $I(\mathbf{x}_1, \ldots, \mathbf{x}_r; \Pi | \mathbf{Z}_1, \ldots, \mathbf{Z}_r) = \Omega\left(nr/(k^2\epsilon^2)\right)$, where $\mathbf{Z}_i = (\mathbf{D}_i, j_i, 0)$ (letting $c_i = 0$ for all i). We claim that $I(\mathbf{x}_t; \Pi | \mathbf{Z}_t, \mathbf{Z}_{-t}, \mathbf{x}_{>t}) \geq CIC_{\mu^n}(L_\infty(k, \epsilon))$. Indeed, the players can hardwire $\mathbf{x}_{>t}$ into the protocol, and Charlie can set $I = t$. Conditioned on $\mathbf{Z}_{-t}$, the inputs to the instances $\mathbf{x}_{<t}$ are independent, and so the players can generate these inputs using their private randomness. Then, for the input of $L_\infty(k, \epsilon)$, the players can embed it as the t-th input to the protocol for the Aug-$L_\infty(k)$ problem. It follows that the output of Aug-$L_\infty(k)$ agrees with the output of $L_\infty(k)$. Moreover, since the distribution on the t-th input instance is μ, we have that

$$I(\mathbf{x}_t; \Pi | \mathbf{Z}_t, \mathbf{Z}_{-t}, \mathbf{x}_{>t}) \geq CIC_{\mu^n}(L_\infty(k, \epsilon)) = \Omega\left(n/(\epsilon^2 k^2)\right)$$

by Theorem 1. It follows that

$$\begin{aligned}
&I(\mathbf{x}_1, \ldots, \mathbf{x}_r; \Pi | \mathbf{Z}_1, \ldots, \mathbf{Z}_r) \\
&= \sum_t I(\mathbf{x}_t; \Pi | \mathbf{Z}_1, \ldots, \mathbf{Z}_r, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_r) \\
&= \sum_t \sum_{z,x} I(\mathbf{x}_t; \Pi | \mathbf{Z}_t, \mathbf{Z}_{-t} = z, \mathbf{x}_{>t} = x) \Pr\{\mathbf{Z}_{-t} = z, \mathbf{x}_{>t} = x\} \\
&\geq \sum_t \sum_{z,x} \Omega\left(\frac{n}{\epsilon^2 k^2}\right) \Pr\{\mathbf{Z}_{-t} = z, \mathbf{x}_{>t} = x\} \\
&= \Omega\left(\frac{nr}{\epsilon^2 k^2}\right)
\end{aligned}$$

as desired. □

4 Frequency Moments

Suppose that $x \in \mathbb{R}^n$. We say that a data stream algorithm solves the (ϵ, p)-Norm problem if its output X satisfies $(1-\epsilon)\|x\|_p^p \leq X \leq (1+\epsilon)\|x\|_p^p$ with probability $\geq 1 - \delta$. Our main result is the following.

Theorem 3. *For any $p > 2$, there exist absolute constants $c > 0$, $\alpha > 1$ and a constant $\epsilon_0 = \epsilon_0(p)$ which depends only on p such that for any $\epsilon \in [c/n^{1/p}, \epsilon_0]$, any randomized streaming algorithm that solves the (ϵ, p)-Norm problem for $x \in \{-M, -M+1, \ldots, M\}^n$ with probability $\geq 19/20$, where $M = \Omega(n^{\alpha/p})$, requires $\Omega(n^{1-2/p}(\log M)/\epsilon^2)$ bits of space.*

Proof. Suppose that a randomized streaming algorithm $\mathcal{A}$ solves the (ϵ, p)-Norm problem with probability $\geq 19/20$. Let $k = \Theta(n^{1/p})$ and $r = (1 - 1/\alpha)\log_{10} M$. We shall reduce the (ϵ, p)-Norm problem to Aug-$L_\infty(r, k, \epsilon)$. Note that with our choice of parameters, $\epsilon k = \Omega(1)$.

Alice generates a stream σ_1 with underlying frequency vector $-\sum_j 10^{j-1} A^j$ and sends the state of $\mathcal{A}$ on σ_1 to Bob. Then Bob generates a stream σ_2 with underlying frequency vector $\sum_j 10^{j-1} B^j$ and continues running $\mathcal{A}$ on σ_2, starting from the state sent by Alice. The streaming algorithm then reaches a state corresponding to an underlying frequency vector $\sum_j 10^{j-1}(B^j - A^j)$. Bob sends this state to Charlie. Charlie, given I and (A^j, B^j) for all $j > I$, generates a stream σ_3 with underlying frequency vector $\sum_{j>I} 10^{j-1}(A^j - B^j)$ and continues running $\mathcal{A}$ on σ_3 to obtain an output V for the execution of $\mathcal{A}$ on a stream with underlying frequency vector $v = \sum_{j=1}^{I} 10^{j-1}(B^j - A^j)$. Finally, Charlie generates a stream σ_4 with underlying frequency vector $10^{I-1} c_I k \mathbf{e}_{j_I}$, where $j_I \in [n]$ and $c_I \in \{0, 1\}$ are the inputs to Charlie in the I-th instance of the $L_\infty(k)$ Promise problem, and continues running $\mathcal{A}$ on σ_4 to obtain an output W for a stream with underlying frequency vector equal to $w = v + 10^{I-1} c_I k \mathbf{e}_{j_I}$.

For notational convenience, let $X^j = B^j - A^j$. We have that $\|X^j\|_\infty \leq 1$ at non-spike positions by the promise of the input to the L_∞ Promise problem.

Notice that $\|v\|_\infty \leq \sum_{j=1}^{I} 10^{j-1} \leq 10^I/9 \leq 10^r/9 \leq M^{1-1/\alpha} \leq M$ and $\|w\|_\infty \leq \|v\|_\infty + k10^{I-1} \leq (k+1)10^r/9 \leq M$ by our assumption of M and choice of k and r. This implies that $\mathcal{A}$ outputs a correct approximation to $\|w\|_\infty$ with probability $\geq 19/20$ and a correct approximation to $\|v\|_\infty$ with probability $\geq 19/20$. Define the event

$$\mathcal{E}_1 = \left\{ \left|W - \|w\|_p^p\right| \leq \epsilon \|w\|_p^p \text{ and } \left|V - \|v\|_p^p\right| \leq \epsilon \|v\|_p^p \right\}.$$

By a union bound, $\Pr\{\mathcal{E}_1\} \geq 9/10$. We show next how to use V and W to solve Aug-$L_\infty(k, r, \epsilon)$, conditioned on the event $\mathcal{E}_1$. Let $L = \left(\sum_{j=1}^{I} 10^{j-1}\right)^p$, then $L = \left((10^I - 1)/9\right)^p \leq (10/9)^p \, 10^{(I-1)p}$.

Case 1. $c_I = 0$ so $w_{j_I} = v_{j_I}$. In this case, $W - V \leq 2\epsilon \|v\|_p^p \leq 2\epsilon n L =: UB_1$,

Case 2. $c_I = 1$ and $w_{j_I} = 10^{I-1}(1-\epsilon)k$. In this case, $v_{j_I} = -10^{I-1}\epsilon k$, so

$$\begin{aligned} W - V &\leq (1+\epsilon)\|w\|_p^p - (1-\epsilon)\|v\|_p^p \\ &= (1+\epsilon)(\|v|_{[n]\setminus\{j_I\}}\|_p^p + \|w_{j_I}\|_p^p) - (1-\epsilon)\|v\|_p^p \\ &= 2\epsilon\|v|_{[n]\setminus\{j_I\}}\|_p^p + (1+\epsilon)\|w|_{j_I}\|_p^p - (1-\epsilon)\|v|_{j_I}\|_p^p \\ &\leq 2\epsilon(n-1)L + (1+\epsilon)10^{(I-1)p}(1-\epsilon)^p k^p - (1-\epsilon)\epsilon^p k^p 10^{(I-1)p} =: UB_2 \end{aligned}$$

and

$$\begin{aligned} W - V &\geq (1-\epsilon)\|w\|_p^p - (1+\epsilon)\|v\|_p^p \\ &= (1-\epsilon)(\|v|_{[n]\setminus\{j_I\}}\|_p^p + \|w|_{j_I}\|_p^p) - (1+\epsilon)\|v\|_p^p \\ &\geq (1-\epsilon)10^{(I-1)p}(1-\epsilon)^p k^p - 2\epsilon(n-1)L - (1+\epsilon)\epsilon^p 10^{(I-1)p} k^p := LB_2 \end{aligned}$$

Case 3. $c_I = 1$ and $w_{j_I} \geq 10^{I-1}k$. In this case, $0 \leq v_{j_I} \leq 10^{I-1}\epsilon k$, so

$$W - V \geq (1-\epsilon)10^{(I-1)p}k^p - 2\epsilon(n-1)L - (1+\epsilon)\epsilon^p 10^{(I-1)p}k^p := LB_3$$

Therefore, Charlie can solve Aug-$L_\infty(r, k, \epsilon)$ provided that $LB_2 > UB_1$ and $LB_3 > UB_2$. It suffices to have

$$\begin{aligned} (1-\epsilon)^{p+1} - (1+\epsilon)\epsilon^p &> 4\epsilon \frac{n}{k^p}\left(\frac{10}{9}\right)^p, \\ \epsilon\left(\frac{p-2}{2} - 2\epsilon^{p-1}\right) &> 4\epsilon\frac{n}{k^p}\left(\frac{10}{9}\right)^p, \end{aligned}$$

which are satisfied when ϵ is small enough, $k = Cn^{1/p}$ for a large enough constant C, and $p > 2$ is a constant. Hence, Charlie can solve the Aug-$L_\infty(r, k, \epsilon)$ problem with probability $\geq 9/10$. The lower bound for the (ϵ, p)-Norm problem follows from Theorem 2. □

5 Lower Bound for Linear Sketches

Given $\eta \geq 0$, define a distribution $\mathcal{D}_{k,\eta}$ on $\mathbb{R}^n$ as follows. Consider $x \sim N(0, I_n)$. Let j be uniformly random in $\{1, \ldots, n\}$. The distribution $\mathcal{D}_{k,\eta}$ is defined to be

$\mathcal{L}(x+(1+\eta)ke_j)$. Suppose that A is an $m \times n$ matrix of orthonormal rows. When operated on vectors $x \sim \mathcal{D}_{k,\eta}$, the product Ax induces a distribution, denoted by $\mathcal{F}_{A,k,\eta}$.

Lemma 6. *Let $\epsilon > 0$. It holds that $d_{TV}(\mathcal{F}_{A,k,0}, \mathcal{F}_{A,k,\epsilon}) \leq \epsilon k\sqrt{m/n}$.*

Proof. Let $y_1 \sim F_{A,k,0}$ and $y_2 \sim F_{A,k,\epsilon}$. By rotational invariance of the Gaussian distribution and the fact that A has orthonormal rows, y_1 is distributed as $x+kA_j$ and y_2 as $x + (1+\epsilon)kA_j$, where $x \sim N(0, I_m)$, A_j is the j-th column of A, and j is uniform on $\{1, \ldots, n\}$.

Suppose the density functions of y_1 and y_2 are $p_1(x)$ and $p_2(x)$ respectively, then

$$p_1(x) = \frac{1}{n}\sum_i p(x - kA_i), \quad p_2(x) = \frac{1}{n}\sum_i p(x - (1+\epsilon)kA_i),$$

where $p(x)$ is the density function of $N(0, I_m)$. It follows that

$$\begin{aligned}
d_{TV}(\mathcal{F}_{A,k,0}, \mathcal{F}_{A,k,\epsilon}) &= \frac{1}{2}\int_{x\in\mathbb{R}^m} |p_1(x) - p_2(x)|dx \\
&= \frac{1}{2}\int_{x\in\mathbb{R}^m} \left|\frac{1}{n}\sum_i p(x - kA_i) - \frac{1}{n}\sum_i p(x - (1+\epsilon)kA_i)\right| dx \\
&\leq \frac{1}{2}\int_{x\in\mathbb{R}^m} \left(\frac{1}{n}\sum_i |p(x - kA_i) - p(x - (1+\epsilon)kA_i|\right) dx \\
&= \frac{1}{n}\sum_i \frac{1}{2}\int_{x\in\mathbb{R}^m} |p(x - kA_i) - p(x - (1+\epsilon)kA_i)|dx \\
&= \frac{1}{n}\sum_i d_{TV}(N(kA_i, I_n) - N((1+\epsilon)kA_i, I_n)) \\
&\leq \frac{1}{n}\sum_i \|kA_i - (1+\epsilon)kA_i\|_2 \qquad \text{(by Proposition 1)} \\
&= \frac{\epsilon k}{n}\sum_i \|A_i\|_2 = \epsilon k \mathbb{E}_j \|A_j\|_2,
\end{aligned}$$

Since $\sum_j \|A_j\|_2^2 = m$, $\mathbb{E}_j\|A_j\|_2^2 = m/n$ and thus $\mathbb{E}\|A_j\|_2 \leq (\mathbb{E}\|A_j\|_2^2)^{1/2} = \sqrt{m/n}$. It follows that $d_{TV}(\mathcal{F}_{A,k,0}, \mathcal{F}_{A,k,\eta}) \leq \epsilon k\sqrt{m/n}$. □

Theorem 4. *Let $p > 2$ be a constant. Consider a distribution over $m \times n$ matrices A for which for every $x \in \mathbb{R}^n$, from Ax one can solve the (ϵ, p)-*Norm *problem, on input x with probability $\geq 3/4$ over the choice of A, where $\epsilon = \Omega(1/n^{1/p})$ is small enough. Then $m = \Omega(n^{1-2/p}/\epsilon^{-2})$.*

Proof. Without loss of generality, we assume that A has orthonormal rows, since we can apply a change of basis to the row space of A in post-processing. Let $k = C_p^{1/p} n^{1/p}$, where C_p is the constant in

$$\mathbb{E}\|z\|_p^p = C_p n, \quad z \sim N(0, I_n).$$

Consider the input x drawn from $\mathcal{D}_0 := \mathcal{D}_{k,0}$ and $\mathcal{D}_1 := \mathcal{D}_{k,2\epsilon}$. Let $b \in \{0,1\}$ indicate that $x \sim \mathcal{D}_b$. We have that $Ax \sim \mathcal{F}_{A,k,0}$ when $b = 0$ and $Ax \sim \mathcal{F}_{A,k,2\epsilon}$ when $b = 1$. Suppose the algorithm outputs W.

Now we compute $\|x\|_p^p$ in each case. When $b = 0$, $\|x\|_p^p = \|x'\|_p^p + |g+k|^p$, where $x' \sim N(0, I_{n-1})$ and $g \sim N(0,1)$ are independent. Since $\|x\|_p$ is a 1-Lipschitz function, by concentration of measure (Lemma 1),

$$\Pr\{|\|x'\|_p - \mathbb{E}\|x'\|_p| \geq 5\} \leq 0.001.$$

Also, $|g| \leq 5$ with probability $\geq 1 - 0.001$. Note that $\mathbb{E}\|x'\|_p^p = C_p(n-1)$. It follows that with probability $\geq 1 - 0.002$, we have

$$\|x\|_p^p \leq 2(1 + o(1))C_p n. \tag{7}$$

Similarly, when $b = 1$, with probability $\geq 1 - 0.002$, it holds that

$$\|x\|_p^p \geq ((1+2\epsilon)^p + 1)(1 - o(1))C_p n. \tag{8}$$

The $o(1)$ in (7) and (8) are of the form $c_p/n^{1/p}$ for some (small) constant $c_p > 0$ that depends only on p. We condition on the event that (7) and (8) hold. With probability $\geq 3/4$, we have

$$\begin{aligned} W &\leq (1+\epsilon)\|x\|_p^p, \quad b = 0 \\ W &\geq (1-\epsilon)\|x\|_p^p, \quad b = 1 \end{aligned}$$

and thus

$$\begin{aligned} W &\leq 2(1+\epsilon)(1+o(1))C_p n, \quad b = 0 \\ W &\geq (1-\epsilon)((1+2\epsilon)^p + 1)(1-o(1))C_p n, \quad b = 1 \end{aligned}$$

So we can recover b from W with probability $\geq 3/4 - 0.002$ provided that

$$\begin{aligned} &2(1+\epsilon)(1+o(1)) < (1-\epsilon)((1+2\epsilon)^p + 1)(1-o(1)) \\ \Longleftarrow &\left(2 + \frac{p+2}{4}\right)\frac{c_p}{n^{1/p}} < \frac{p-2}{2}\epsilon \qquad \text{(recall that } o(1) \text{ is actually } c_p/n^{1/p}) \end{aligned}$$

which holds for ϵ small enough while satisfying that $\epsilon = \Omega(1/n^{1/p})$. Consider the event $\mathcal{E}$ that the algorithm's output indicates $b = 1$. Then $\Pr(\mathcal{E}|x \sim \mathcal{D}_0) \leq 1/4 + 0.002$ while $\Pr(\mathcal{E}|x \sim \mathcal{D}_1) \geq 3/4 - 0.002$. By definition of total variation distance,

$$d_{TV}(\mathcal{F}_{A,k,0}, \mathcal{F}_{A,k,2\epsilon}) \geq |\Pr(\mathcal{E}|x \sim \mathcal{D}_1) - \Pr(\mathcal{E}|x \sim \mathcal{D}_0)| \geq 1/2 + 0.004.$$

On the other hand, by the preceding lemma, $d_{TV}(\mathcal{F}_{A,k,0}, \mathcal{F}_{A,k,2\epsilon}) \leq 2\epsilon k\sqrt{m/n}$. Therefore it must hold that $m = \Omega\left(n/(k^2\epsilon^2)\right) = \Omega\left(n^{1-2/p}/\epsilon^2\right)$. □

References

1. Indyk, P.: Sketching, streaming and sublinear-space algorithms (2007), Graduate course notes available at `http://stellar.mit.edu/S/course/6/fa07/6.895/`
2. Muthukrishnan, S.: Data Streams: Algorithms and Applications. Foundations and Trends in Theoretical Computer Science 1(2), 117–236 (2005)
3. Alon, N., Matias, Y., Szegedy, M.: The Space Complexity of Approximating the Frequency Moments. JCSS 58(1), 137–147 (1999)
4. Flajolet, P., Martin, G.N.: Probabilistic counting. In: Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 76–82 (1983)
5. Chakrabarti, A., Do Ba, K., Muthukrishnan, S.: Estimating Entropy and Entropy Norm on Data Streams. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 196–205. Springer, Heidelberg (2006)
6. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 693–703. Springer, Heidelberg (2002)
7. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. J. Algorithms 55(1), 58–75 (2005)
8. Indyk, P., Woodruff, D.P.: Optimal approximations of the frequency moments of data streams. In: STOC, pp. 202–208 (2005)
9. Bhuvanagiri, L., Ganguly, S., Kesh, D., Saha, C.: Simpler algorithm for estimating frequency moments of data streams. In: SODA, pp. 708–713 (2006)
10. Monemizadeh, M., Woodruff, D.P.: 1-pass relative-error ℓ_p-sampling with applications. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, pp. 1143–1160 (2010)
11. Andoni, A., Krauthgamer, R., Onak, K.: Streaming algorithms via precision sampling. In: FOCS, pp. 363–372 (2011)
12. Braverman, V., Ostrovsky, R.: Recursive sketching for frequency moments. CoRR abs/1011.2571 (2010)
13. Andoni, A.: High frequency moment via max stability, `http://web.mit.edu/andoni/www/papers/fkStable.pdf`
14. Ganguly, S.: Polynomial estimators for high frequency moments. CoRR abs/1104.4552 (2011)
15. Woodruff, D.P.: Optimal space lower bounds for all frequency moments. In: Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 167–175 (2004)
16. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: An information statistics approach to data stream and communication complexity. J. Comput. Syst. Sci. 68(4), 702–732 (2004)
17. Chakrabarti, A., Khot, S., Sun, X.: Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In: CCC, pp. 107–117 (2003)
18. Woodruff, D.P., Zhang, Q.: Tight bounds for distributed functional monitoring. In: Proceedings of the 44th Symposium on Theory of Computing, STOC 2012, pp. 941–960 (2012)
19. Ganguly, S.: A lower bound for estimating high moments of a data stream. CoRR abs/1201.0253 (2012)
20. Braverman, V., Ostrovsky, R.: Approximating large frequency moments with pick-and-drop sampling. CoRR abs/1212.0202 (2012)

21. Coppersmith, D., Kumar, R.: An improved data stream algorithm for frequency moments. In: Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 151–156 (2004)
22. Ganguly, S.: Estimating frequency moments of data streams using random linear combinations. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) APPROX and RANDOM 2004. LNCS, vol. 3122, pp. 369–380. Springer, Heidelberg (2004)
23. Ganguly, S.: A hybrid algorithm for estimating frequency moments of data streams. Manuscript (2004)
24. Andoni, A., Nguy˜ên, H.L., Polyanskiy, Y., Wu, Y.: Tight lower bound for linear sketches of moments. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 25–32. Springer, Heidelberg (2013)
25. Price, E., Woodruff, D.P.: Applications of the shannon-hartley theorem to data streams and sparse recovery. In: ISIT, pp. 2446–2450 (2012)
26. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D., Trevisan, L.: Counting distinct elements in a data stream. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 1–10. Springer, Heidelberg (2002)
27. Chakrabarti, A., Cormode, G., McGregor, A.: A near-optimal algorithm for estimating the entropy of a stream. ACM Transactions on Algorithms 6(3) (2010)
28. Clarkson, K.L., Woodruff, D.P.: Numerical linear algebra in the streaming model. In: STOC, pp. 205–214 (2009)
29. Ganguly, S.: Lower bounds on frequency estimation of data streams (extended abstract). In: CSR, pp. 204–215 (2008)
30. Ganguly, S.: Deterministically estimating data stream frequencies. In: Du, D.-Z., Hu, X., Pardalos, P.M. (eds.) COCOA 2009. LNCS, vol. 5573, pp. 301–312. Springer, Heidelberg (2009)
31. Ganguly, S., Cormode, G.: On estimating frequency moments of data streams. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) APPROX and RANDOM 2007. LNCS, vol. 4627, pp. 479–493. Springer, Heidelberg (2007)
32. Indyk, P., Woodruff, D.P.: Tight lower bounds for the distinct elements problem. In: FOCS, pp. 283–288 (2003)
33. Kane, D.M., Nelson, J., Porat, E., Woodruff, D.P.: Fast moment estimation in data streams in optimal space. In: STOC, pp. 745–754 (2011)
34. Pavan, A., Tirthapura, S.: Range-efficient counting of distinct elements in a massive data stream. SIAM J. Comput. 37(2), 359–379 (2007)
35. Chakrabarti, A., Shi, Y., Wirth, A., Yao, A.C.C.: Informational complexity and the direct sum problem for simultaneous message complexity. In: FOCS, pp. 270–278 (2001)
36. Cormode, G., Muthukrishnan, S.: Space efficient mining of multigraph streams. In: PODS, pp. 271–282 (2005)
37. Jayram, T.S., Woodruff, D.P.: The data stream space complexity of cascaded norms. In: FOCS, pp. 765–774 (2009)
38. Ledoux, M., Talagrand, M.: Probability in Banach Spaces: Isoperimetry and Processes. Springer (1991)
39. DasGupta, A.: Asymptotic Theory of Statistics and Probability. Springer (2008)
40. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press (1997)
41. Bar-Yossef, Z.: The Complexity of Massive Data Set Computations. PhD thesis, University of California, Berkeley (2002)

Improved FPTAS for Multi-spin Systems

Pinyan Lu[1] and Yitong Yin[2,⋆]

[1] Microsoft Research Asia, China
pinyanl@microsoft.com
[2] State Key Laboratory for Novel Software Technology, Nanjing University, China
yinyt@nju.edu.cn

Abstract. We design deterministic fully polynomial-time approximation scheme (FPTAS) for computing the partition function for a class of multi-spin systems, extending the known approximable regime by an exponential scale. As a consequence, we have an FPTAS for the Potts models with inverse temperature β up to a critical threshold $|\beta| = O(\frac{1}{\Delta})$ where Δ is the maximum degree, confirming a conjecture in [10]. We also give an improved FPTAS for a generalization of counting q-colorings, namely the counting list-colorings. As a consequence we have an FPTAS for counting q-colorings in graphs with maximum degree Δ when $q \geq \alpha\Delta + 1$ for α greater than $\alpha^* \approx 2.58071$. This is so far the best bound achieved by deterministic approximation algorithms for counting q-colorings. All these improvements are obtained by applying a potential analysis to the correlation decay on computation trees for multi-spin systems.

1 Introduction

Spin systems in Statistical Physics are the stochastic models defined by local interactions. In Computer Science, spin systems are used as a theoretical framework for counting or inference problems arising from constraint satisfaction problems, e.g. counting independent sets or q-colorings in graphs, and probability inference in graphical models.

A central problem in this framework is the computation of the *partition function*, which may solve both counting and inference. The problem is #P-hard for almost all nontrivial spin systems [3, 4]. A classic approach for approximation of partition function is the Markov Chain Monte Carlo (MCMC) method which relies on the rapid mixing of random walks in the configuration space [5–7,13–18,21,27]. A more contemporary approach is the correlation decay technique introduced by Bandyopadhyay and Gamarnik [1] and Weitz [28], which leads to deterministic fully polynomial-time approximation scheme (FPTAS) for #P-hard counting problems [2, 10, 19, 20, 22, 23, 29].

In these algorithms, the computation of a marginal probability (which is equivalent to the computation of partition function by self-reduction) is reduced to

⋆ Supported by the National Science Foundation of China under Grant No. 61272081, 61003023 and 61021062.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 639–654, 2013.

evaluating an exponential-size tree-structured dynamical system. The correlation decay property guarantees that the far-away variables can be disregarded without substantially affecting the marginal probability of interest, thus the true values can be efficiently approximated by evaluating truncated dynamical systems. Two such dynamical systems were proposed: (1) the self-avoiding-walk (SAW) tree [28] for two-state spin systems and (2) the computation tree [10] for all spin systems. For two-state spin systems, FPTAS based on SAW-trees may approach the approximability boundaries, such as [19,20,23,28]. This is because a SAW-tree is a faithful construction of the original spin system on a tree, hence long-range correlations can be used as gadgets in the reduction for the inapproximability [8,25,26]. Very recently, the similar long-range correlations were used to prove inapproximability for multi-spin systems [9]. On the algorithmic side, due to a barrier result of Sly in [24], the original spin systems on trees are no longer capable of simulating all marginal probabilities. The computation tree introduced by Gamarnik and Katz in [10] overcomes this fundamental issue by creating a dynamical system consisting of different instances of spin systems.

1.1 Our Results

We design efficient computation trees for multi-spin systems: For a vertex of degree d, our computation tree expends to d branches while the previous one in [10] has $\exp(\Omega(d))$ many branches. We apply a potential analysis to the decay of correlation between variables in computation trees. The potential analysis has been used in [19,20,22,23] for analyzing the correlation decay on the self-avoiding walk trees for two-state spin systems. We show for the first time that this powerful technique can be applied to computation trees for multi-spin systems. Our new construction of efficient computation trees and potential analysis greatly extend the regimes of correlation decay and deterministic FPTAS for these systems.

One of the most well-studied multi-spin systems is the Potts model.

Theorem 1. *For any constant $q \geq 2$, there exists an FPTAS for computing the partition function for q-state Potts models with inverse temperature β and maximum degree Δ satisfying $3\Delta(\mathrm{e}^{|\beta|} - 1) \leq 1$.*

For large Δ, the condition $3\Delta(\mathrm{e}^{|\beta|} - 1) \leq 1$ is translated to that $|\beta| = O(\frac{1}{\Delta})$, which greatly improves the best previous bound $\beta = O\left(\frac{1}{\Delta q^{\Delta}}\right)$ due to Gamarnik and Katz [10] and also confirms a conjecture in [10]. For the anti-ferromagnetic case ($\beta < 0$), our condition is asymptotically tight due to a very recent inapproximability result of Galanis, Štefankovič, and Vigoda [9].

Theorem 1 is a special case of a much more general theorem for the q-state spin systems, also called the Markov random fields. As suggested by [10], the regime of correlation decay for these models is described in terms of $c_{\boldsymbol{A}}$, the maximum ratio between edge parameters. We show that there exists an FPTAS for a family of Markov random fields if $3\Delta(c_{\boldsymbol{A}} - 1) \leq 1$. This exponentially improves the previous best known condition $(c_{\boldsymbol{A}}^{\Delta} - c_{\boldsymbol{A}}^{-\Delta})\Delta q^{\Delta} < 1$ proved in [10]. This general result is formally stated as Theorem 3 in Section 2.

We next study the problem of counting proper q-colorings in an undirected graph. For this problem, the mixing or the tractability condition is usually given in form of $q \geq \alpha\Delta + \beta$ for some constant α and β where Δ is the maximum degree of the graph. The previous best bound for deterministic FPTAS was achieved in [10] for an $\alpha \approx 2.8432$ and some sufficiently large β on triangle-free graphs. Better bounds (with $\alpha < 2$) were known for randomized approximation algorithms [6, 15, 27] or correlation decay only [11, 12]. We prove the following theorem for a constant $\alpha^* \approx 2.58071$ which is formally defined by (2) in Section 2.

Theorem 2. *There exists an FPTAS for counting q-colorings on graphs with maximum degree Δ if q and Δ are constants and $q \geq \alpha\Delta + 1$ for $\alpha > \alpha^*$.*

This is a new record for the deterministic FPTAS for counting q-colorings on general graphs, and we remove the triangle-free requirement in previous correlation-decay based results such as [10, 12]. Theorem 2 is proved as a special case of a theorem for a generalization of q-colorings, called the list-colorings, which is formally stated as Theorem 4 in Section 2.

All the above FPTAS require the degree of the graph and the number of states (colors) to be constant. If we remove this restriction, the algorithms compute a $(1 \pm \epsilon)$-approximation of the true value for any fixed $0 < \epsilon < 1$ in time $n^{O(\log n)}$. This complexity bound was only known previously for simple models like list-colorings but was not known for general multi-spin systems since for such systems the previous computation tree proposed in [10] tries to enumerate all configurations of the local neighborhood at each step. We give a more efficient computation tree which uses exponentially less branches.

2 Definitions and Statements of Results

An instance of a *q-state spin system* or a pair-wise *Markov random field (MRF)* is a tuple $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F})$, where

- $G = (V, E)$ is an undirected graph called the *underlying graph*;
- $\mathcal{X} = [q] = \{1, 2, \ldots, q\}$ is a domain of *spin states*;
- $\boldsymbol{A} = (A_e, e \in E)$ is a tuple where each $A_e : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a symmetric function specifying the *activity* of edge e;
- $\boldsymbol{F} = (F_v, v \in V)$ is a tuple where each $F_v : \mathcal{X} \to \mathbb{R}_{\geq 0}$ specifies the *external field* at vertex v.

The size of an MRF instance is defined as $|\Omega| = \max\{|V|, |\mathcal{X}|\}$. We consider only those MRF instances such that the number of bits used to encode $\boldsymbol{A}$ and $\boldsymbol{F}$ is in polynomial of $n = |V|$ and $q = |\mathcal{X}|$. This does not affect the generality of the problem since we are interested in the approximation algorithms.

The *partition function* of an MRF instance $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F})$ is defined as

$$Z(\Omega) \triangleq \sum_{\boldsymbol{x} \in \mathcal{X}^V} \prod_{e=uv \in E} A_e(x_u, x_v) \prod_{v \in V} F_v(x_v).$$

This gives rise to a probability distribution $\mathbb{P}_\Omega$, called the *Gibbs measure*, over all configurations $\boldsymbol{x} \in \mathcal{X}^V$, such that

$$\mathbb{P}_\Omega(\boldsymbol{X} = \boldsymbol{x}) = \frac{\prod_{e=uv\in E} A_e(x_u, x_v) \prod_{v\in V} F_v(x_v)}{Z(\Omega)}.$$

Given an MRF instance $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F})$ with underlying graph $G = (V, E)$, we denote by Δ_G the maximum degree of G and let

$$c_{\boldsymbol{A}} \triangleq \max_{\substack{e\in E \\ w,x,y,z\in\mathcal{X}}} \frac{A_e(x,y)}{A_e(w,z)}.$$

Theorem 3. *Let $\mathbb{M}$ be a family of MRF instances with bounded degree and bounded size of domain. There exists an FPTAS for computing the partition function of MRFs in $\mathbb{M}$ if it holds that*

$$\forall \Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F}) \in \mathbb{M}, \qquad 3\Delta_G(c_{\boldsymbol{A}} - 1) \le 1. \tag{1}$$

For any family $\mathbb{M}$ of MRFs satisfying (1) *without any restriction on the degree or the size of domain, the algorithm computes a $(1 \pm \epsilon)$-approximation of $Z(\Omega)$ for any fixed $0 < \epsilon < 1$ in time $n^{O(\log n)}$ where $n = |\Omega|$.*

The q-state Potts model is a special class of MRFs $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F})$ with for every $e \in E$, $A_e = A$ such that $A(x, y) = \mathrm{e}^\beta$ if $x = y$ and $A(x, y) = 1$ otherwise. The parameter β is called the *inverse temperature*. It is easy to see that Theorem 1 is a special case of Theorem 3 on Potts models.

Next we consider the proper q-colorings in an undirected graph, which can be easily seen as a special case of MRF. The problem of counting q-colorings is solved by solving its generalization called the list-colorings. A list-coloring instance is a tuple $\Omega = (G, \mathcal{X}, \boldsymbol{L})$ with that

- $G = (V, E)$ is an undirected graph;
- $\mathcal{X} = [q]$ is a domain of q colors;
- $\boldsymbol{L} = (L_v, v \in V)$ such that each $L_v \subseteq \mathcal{X}$ is a list of colors for vertex v.

A proper coloring in a list-coloring instance is a proper q-coloring $\boldsymbol{x} \in \mathcal{X}^V$ of vertices such that $x_v \in L_v$ for every $v \in V$. The list-coloring is a special case of MRFs $(G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F})$ with that for every $e \in E$, $A_e = A$ such that $A(x, y) = 0$ if $x = y$ and $A(x, y) = 1$ if otherwise, and for every $v \in V$, the external field F_v is a Boolean function indicating the color list L_v. For the list-colorings we have $c_{\boldsymbol{A}} = \infty$, thus Theorem 3 does not apply, so we use different algorithm and analysis to prove the following theorem. Let $\alpha^* \approx 2.58071$ be the solution to the equation[1]

$$\frac{\sqrt{2} + \sqrt{2\alpha - 1 - \sqrt{4\alpha - 3}}}{\sqrt{2\alpha(\alpha - 1)}} \exp\left(\frac{3 - 2\alpha + \sqrt{4\alpha - 3}}{4(\alpha - 1)}\right) = 1. \tag{2}$$

[1] The LHS of (2) is in fact monotonously decreasing from $+\infty$ to 0 for $\alpha > 1$, so there is a unique solution α^*.

Theorem 4. *There exists a deterministic FPTAS for counting proper colorings in list-coloring instances $\Omega = (G, \mathcal{X}, \boldsymbol{L})$ with bounded degree Δ_G and bounded number of colors $q = |\mathcal{X}|$ satisfying that there is an $\alpha > \alpha^*$ such that*

$$\forall v \in V, \quad |L_v| \geq \alpha \Delta_G + 1. \tag{3}$$

Obviously Theorem 2 is a special case of Theorem 4 as colorings are just list-clorings with $L_v = \mathcal{X}$ for every $v \in V$.

3 Markov Random Fields

Given an MRF instance defined on the underlying graph $G = (V, E)$, we suppose that for each vertex $v \in V$, the neighbors of v are enumerated as $v_1, v_2, \ldots, v_{\deg(v)}$ where $\deg(v)$ is the degree of v. We define operations called *pinning* and *partial pinning* on MRF instances as follows.

Definition 1. *Given an MRF instance $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F})$, a vertex $v \in V$ and its neighbors $v_1, v_2, \ldots, v_d$ in G, where $d = \deg(v)$, for each spin state $x \in \mathcal{X}$ and each $1 \leq i \leq d+1$, the* partial pinning *of Ω, denoted as $\Omega^i_{v,x}$, is a new MRF instance augmented from Ω as $\Omega^i_{v,x} = (G_v, \mathcal{X}, \widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{F}})$, where $G_v = G \setminus \{v\}$ is the subgraph of G induced by $V \setminus \{v\}$, $\widetilde{\boldsymbol{A}} = (A_e, e \in E \setminus \{vv_1, vv_2, \ldots, vv_d\})$ is the restriction of $\boldsymbol{A}$ on the set of edges in G_v, and $\widetilde{\boldsymbol{F}} = (\widetilde{F}_u, u \in V \setminus \{v\})$ where*

$$\forall y \in \mathcal{X}, \quad \widetilde{F}_u(y) = \begin{cases} A_{uv}(x, y) F_u(y) & \text{if } u \in \{v_1, \ldots, v_{i-1}\}, \\ F_u(y) & \text{otherwise.} \end{cases}$$

The pinning *of Ω is a partial pinning by choosing $i = d+1$, which is denoted as $\Omega_{v,x} = \textsc{Pin}_{v,x}(\Omega) = \Omega^{d+1}_{v,x}$.*

The following identity can be seen as a generalization of the recursion for list-colorings derived in [10]. Compared to the recursion for MRFs in [10], it uses substantially less variables.

Proposition 1. *Let $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F})$ be an MRF instance. For every vertex $v \in V_G$ and its neighbors $v_1, v_2, \ldots, v_d$ where $d = \deg(v)$, and every spin state $x \in \mathcal{X}$, it holds that*

$$\mathbb{P}_\Omega(X_v = x) = \frac{F_v(x) \prod_{i=1}^d \Big(A_{vv_i}(x,x) - \sum_{z \neq x} \big(A_{vv_i}(x,x) - A_{vv_i}(x,z) \big) \mathbb{P}_{\Omega^i_{v,x}}(X_{v_i} = z) \Big)}{\sum_{y \in \mathcal{X}} F_v(y) \prod_{i=1}^d \Big(A_{vv_i}(y,y) - \sum_{z \neq y} \big(A_{vv_i}(y,y) - A_{vv_i}(y,z) \big) \mathbb{P}_{\Omega^i_{v,y}}(X_{v_i} = z) \Big)}.$$

Proof. We define that

$$Z_\Omega(X_v = x) \triangleq \sum_{\substack{\boldsymbol{x} \in \mathcal{X}^V \\ x_v = x}} \prod_{uw \in E} A_{uw}(x_u, x_w) \prod_{u \in V} F_u(x_u).$$

It can be verified that $Z_\Omega(X_v = x) = F_v(x)Z(\Omega_{v,x})$ where $\Omega_{v,x} = \textsc{Pin}_{v,x}(\Omega)$ is the pinning of Ω. Then

$$\mathbb{P}_\Omega(X_v = x) = \frac{Z_\Omega(X_v = x)}{\sum_{y\in\mathcal{X}} Z_\Omega(X_v = y)} = \frac{F_v(x)Z(\Omega_{v,x})}{\sum_{y\in\mathcal{X}} F_v(y)Z(\Omega_{v,y})}. \tag{4}$$

By the Definition 1, it holds that $\Omega_{v,x} = \Omega^{d+1}_{v,x}$, and $\Omega^1_{v,x}$ is simply the MRF instance deleting vertex v, which is independent of the choice of x. Therefore,

$$(4) = \frac{F_v(x)Z(\Omega^{d+1}_{v,x})/Z(\Omega^1_{v,x})}{\sum_{y\in\mathcal{X}} F_v(y)Z(\Omega^{d+1}_{v,y})/Z(\Omega^1_{v,y})} = \frac{F_v(x)\prod_{i=1}^d \frac{Z(\Omega^{i+1}_{v,x})}{Z(\Omega^i_{v,x})}}{\sum_{y\in\mathcal{X}} F_v(y)\prod_{i=1}^d \frac{Z(\Omega^{i+1}_{v,y})}{Z(\Omega^i_{v,y})}}. \tag{5}$$

The partition function of a partial pinning of Ω expands as:

$$Z(\Omega^i_{v,x}) = \sum_{\boldsymbol{x}\in\mathcal{X}^{V\setminus\{v\}}} \prod_{\substack{uw\in E\\ u\neq v\\ w\neq v}} A_{uw}(x_u, x_w) \prod_{u\in V\setminus\{v\}} F_u(x_u) \prod_{j=1}^{i-1} A_{vv_j}(x, x_{v_j}).$$

It can be verified that $Z(\Omega^{i+1}_{v,x}) = \sum_{z\in\mathcal{X}} A_{vv_i}(x,z)\cdot Z_{\Omega^i_{v,x}}(X_{v_i} = z)$. Therefore,

$$\begin{aligned}
(5) &= \frac{F_v(x)\prod_{i=1}^d \sum_{z\in\mathcal{X}} A_{vv_i}(x,z)\cdot \frac{Z_{\Omega^i_{v,x}}(X_{v_i}=z)}{Z(\Omega^i_{v,x})}}{\sum_{y\in\mathcal{X}} F_v(y)\prod_{i=1}^d \sum_{z\in\mathcal{X}} A_{vv_i}(y,z)\cdot \frac{Z_{\Omega^i_{v,y}}(X_{v_i}=z)}{Z(\Omega^i_{v,y})}} \\
&= \frac{F_v(x)\prod_{i=1}^d \sum_{z\in\mathcal{X}} A_{vv_i}(x,z)\cdot \mathbb{P}_{\Omega^i_{v,x}}(X_{v_i} = z)}{\sum_{y\in\mathcal{X}} F_v(y)\prod_{i=1}^d \sum_{z\in\mathcal{X}} A_{vv_i}(y,z)\cdot \mathbb{P}_{\Omega^i_{v,y}}(X_{v_i} = z)} \\
&= \frac{F_v(x)\prod_{i=1}^d \Big(A_{vv_i}(x,x) - \sum_{z\neq x}\big(A_{vv_i}(x,x) - A_{vv_i}(x,z)\big)\mathbb{P}_{\Omega^i_{v,x}}(X_{v_i}=z)\Big)}{\sum_{y\in\mathcal{X}} F_v(y)\prod_{i=1}^d \Big(A_{vv_i}(y,y) - \sum_{z\neq y}\big(A_{vv_i}(y,y) - A_{vv_i}(y,z)\big)\mathbb{P}_{\Omega^i_{v,y}}(X_{v_i}=z)\Big)},
\end{aligned}$$

where the last equation uses the fact that $\sum_{z\in\mathcal{X}} \mathbb{P}_{\Omega^i_{v,y}}(X_{v_i} = z) = 1$.

3.1 Algorithms Based on the Computation Tree Recursion

Given an MRF instance $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F})$ on underlying graph $G = (V, E)$, a vertex $v \in V$ with d neighbors $v_1, v_2, \ldots, v_d$ in G and a spin state $x \in \mathcal{X}$, we define the following function:

$$f_{\Omega,v,x}(\boldsymbol{p}) \triangleq \frac{F_v(x)\prod_{i=1}^d \big(A_{vv_i}(x,x) - \sum_{z\neq x}\big(A_{vv_i}(x,x) - A_{vv_i}(x,z)\big)p_{i,x,z}\big)}{\sum_{y\in\mathcal{X}} F_v(y)\prod_{i=1}^d \big(A_{vv_i}(y,y) - \sum_{z\neq y}\big(A_{vv_i}(y,y) - A_{vv_i}(y,z)\big)p_{i,y,z}\big)} \tag{6}$$

over the domain of vectors $\boldsymbol{p} = (p_{i,y,z}, 1 \le i \le d; y, z \in \mathcal{X}; y \neq z) \in [0,1]^{dq(q-1)}$ satisfying that $\sum_{z\neq y} p_{i,y,z} \le 1$ for every $1 \le i \le d$ and $y \in \mathcal{X}$. Due to Proposition 1 we have $\mathbb{P}_\Omega(X_v = x) = f_{\Omega,v,x}(\boldsymbol{p})$ where $p_{i,y,z} = \mathbb{P}_{\Omega^i_{v,y}}(X_{v_i} = z)$ for each $1 \le i \le d$ and $y, z \in \mathcal{X}$ that $y \neq z$. This already gives us a procedure, called

the *computation tree recursion*, for computing the exact value of a marginal probability $\mathbb{P}_{\Omega}(X_v = x)$. Note that it terminates since each partial pinning $\Omega^i_{v,y}$ deletes a vertex v from the current underlying graph.

It is easy to verify the following closure property of the computation tree recursion: If each $p_{i,y,z}$ is replaced by an estimation $\widehat{\mathbb{P}}_{\Omega^i_{v,y}}(X_{v_i} = z)$ of marginal $\mathbb{P}_{\Omega^i_{v,y}}(X_{v_i} = z)$ such that $\widehat{\mathbb{P}}_{\Omega^i_{v,y}}(X_{v_i} = z) \in [0,1]$ and $\sum_{z \neq y} \widehat{\mathbb{P}}_{\Omega^i_{v,y}}(X_{v_i} = z) \leq 1$ then the outcome of the recursion $\widehat{\mathbb{P}}_{\Omega}(X_v = x) = f_{\Omega,v,x}(\boldsymbol{p})$ as an estimation of $\mathbb{P}_{\Omega}(X_v = x)$ still satisfies that $\widehat{\mathbb{P}}_{\Omega}(X_v = x) \in [0,1]$ and $\sum_{x \in \mathcal{X}} \widehat{\mathbb{P}}_{\Omega}(X_v = x) = 1$.

The size of the computation tree can be easily of exponential in the size of the underlying graph. We can run the computation tree recursion up to t levels and use a naive estimation of marginals for the base cases. Formally, for $t \geq 0$, the quantity $\widehat{\mathbb{P}}^{(t)}_{\Omega}(X_v = x)$ is recursively defined as follows:

- If $t = 0$, let $\widehat{\mathbb{P}}^{(0)}_{\Omega}(X_v = x) = \frac{F_v(x)}{\sum_{y \in \mathcal{X}} F_v(y)}$.
- If $t > 0$, let $\widehat{\mathbb{P}}^{(t)}_{\Omega}(X_v = x) = f_{\Omega,v,x}(\widehat{\boldsymbol{p}})$ where $\hat{p}_{i,y,z} = \widehat{\mathbb{P}}^{(t-1)}_{\Omega^i_{v,y}}(X_{v_i} = z)$ for each $1 \leq i \leq d$ and $y, z \in \mathcal{X}$ that $y \neq z$.

The value of the base case $\widehat{\mathbb{P}}^{(0)}_{\Omega}(X_v = x)$ is not important due to a correlation decay property we prove later. As shown in [10], on graphs of constant maximum degrees, the quantity $\widehat{\mathbb{P}}^{(t)}_{\Omega}(X_v = x)$ can be efficiently computed by dynamic programming when $t = O(\log n)$.

The partition function can be approximated from estimations of marginals by the following standard procedure. Enumerate the vertices in V as $v_1, v_2, \ldots, v_n$.

1. Let $\Omega_1 = \Omega$. For $k = 1, 2, \ldots, n$, assuming that the Ω_k is well-defined, use the computation tree recursion to compute $\widehat{\mathbb{P}}^{(t)}_{\Omega_k}(X_{v_k} = x)$ for all $x \in \mathcal{X}$, choose x_k to be the x which maximizes the $\widehat{\mathbb{P}}^{(t)}_{\Omega_k}(X_{v_k} = x)$ and construct $\Omega_{k+1} = \textsc{Pin}_{v_k,x_k}(\Omega_k)$ as a pinning of Ω_k.
2. Compute that $\widehat{Z}(\Omega) = \frac{\prod_{e=uv \in E} A_e(x_u, x_v) \prod_{v \in V} F_v(x_v)}{\prod_{k=1}^{n} \widehat{\mathbb{P}}^{(t)}_{\Omega_k}(X_{v_k} = x_{v_k})}$ and return $\widehat{Z}(\Omega)$.

This algorithm is the same as the one proposed in [10], except for using a simplified computation tree recursion, thus by the same analysis as in [10], we have the following proposition.

Proposition 2. *Let $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F})$ be an MRF instance such that G has maximum degree Δ and $q = |\mathcal{X}|$. The value of $\widehat{Z}(\Omega)$ can be computed in time* $\mathrm{poly}(|\Omega|) \cdot (q\Delta)^{O(t)}$.

3.2 Correlation Decay on the Computation Tree

The above algorithm approximates the marginal probabilities by simulating a tree-structured dynamical system for a limited number of iterations. The accuracy of this approximation relies on the following property of correlation decay.

Definition 2 (Correlation Decay). *Let $\mathbb{M}$ be a family of MRFs. We say that the computation tree recursion exhibits* exponential correlation decay *over $\mathbb{M}$ if there exists a constant $C > 0$ such that given any MRF instance $\Omega \in \mathbb{M}$, for all $t \geq 1$, it holds that*

$$\max_{\substack{v \in V_\Omega \\ x \in \mathcal{X}}} \left| \mathbb{P}_\Omega(X_v = x) - \widehat{\mathbb{P}}_\Omega^{(t)}(X_v = x) \right| \leq \mathrm{poly}(|\Omega|) \cdot \exp(-C \cdot t).$$

A sufficient condition for the exponential correlation decay is that the error of estimation decays by a constant factor in every iteration. However, in general, the systems exhibiting correlation decay may not necessarily decay in every step. This issue has been addressed by a potential-based analysis in [19, 20, 22, 23] for self-avoiding walk trees for 2-spin systems, which is now formalized as the following condition for computation trees for multi-spin systems.

Definition 3 (The Amortized Decay Condition). *Let $\mathbb{M}$ be a family of q-state MRFs. We say that $\mathbb{M}$ satisfies the* Amortized Decay Condition *if there exists a strictly increasing differentiable function $\varphi : [0, 1] \to \mathbb{R}$ satisfying the following conditions:*

1. *Let $\Phi(x) = \frac{\mathrm{d}\,\varphi(x)}{\mathrm{d}\,x}$ denote the derivative of function φ. We call $\Phi(\cdot)$ the* potential function. *The values of $\Phi(\cdot)$ and $\frac{1}{\Phi(\cdot)}$ are bounded by* $\mathrm{poly}(q)$ *over domain $[0, 1]$.*
2. *Given an MRF instance $\Omega \in \mathbb{M}$, a vertex $v \in V_\Omega$ with $d = \deg(v)$ and a spin state $x \in \mathcal{X}$, let $f = f_{\Omega,v,x}$ be the computation tree recursion defined by* (6)*, and define the* amortized decay rate *as*

$$\kappa(\boldsymbol{p}) \triangleq \sum_{\substack{1 \leq i \leq d \\ y \neq z}} \left| \frac{\partial f(\boldsymbol{p})}{\partial p_{i,y,z}} \right| \frac{\Phi(f(\boldsymbol{p}))}{\Phi(p_{i,y,z})}. \tag{7}$$

There exists a constant $0 < \kappa < 1$, such that for every MRF instance $\Omega \in \mathbb{M}$, vertex $v \in V_\Omega$ and spin state $x \in \mathcal{X}$, it holds that $\kappa(\boldsymbol{p}) \leq \kappa$ for all $\boldsymbol{p} = (p_{i,y,z}, 1 \leq i \leq d \wedge y, z \in \mathcal{X} \wedge y \neq z) \in [0, 1]^{dq(q-1)}$ satisfying that $\sum_{z \neq y} p_{i,y,z} \leq 1$ for all i and y.

We may replace the first condition by a more sophisticated bound on the values of $|\Phi(\cdot)|$ and $\frac{1}{|\Phi(\cdot)|}$, which will give us more freedom to choose potential functions, although the current simple bound is sufficient for our analysis.

We say a family $\mathbb{M}$ of MRF instances is *closed under partial pinning* if for every $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F}) \in \mathbb{M}$, every vertex $v \in V_G$ with $d = \deg(v)$, spin state $x \in \mathcal{X}$ and $1 \leq i \leq d$, it holds for the partial pinning $\Omega^i_{v,x}$ of Ω that $\Omega^i_{v,x} \in \mathbb{M}$.

Lemma 1. *Let $\mathbb{M}$ be a family of MRFs which is closed under partial pinning. If $\mathbb{M}$ satisfies the amortized decay condition then the computation tree recursion exhibits exponential correlation decay over $\mathbb{M}$.*

Proof. Pick an MRF instance $\Omega \in \mathbb{M}$, a vertex $v \in V_\Omega$ with d neighbors $v_1, v_2, \ldots, v_d$ and a spin state $x \in \mathcal{X}$. Let $\varphi : [0, 1] \to \mathbb{R}$ be the monotone

differentiable function and $\Phi(\cdot)$ be its derivative, as required by the amortized decay condition. Consider the corresponding recursion $f = f_{\Omega,v,x}$.

We define the following notations: Let $p = \mathbb{P}_{\Omega}(X_v = x)$, $\hat{p} = \widehat{\mathbb{P}}_{\Omega}^{(t)}(X_v = x)$, and for every $1 \le i \le d$ and $y, z \in \mathcal{X}$ that $y \neq z$, let $p_{i,y,z} = \mathbb{P}_{\Omega^i_{v,y}}(X_{v_i} = z)$ and $\hat{p}_{i,y,z} = \widehat{\mathbb{P}}^{(t-1)}_{\Omega^i_{v,y}}(X_{v_i} = z)$. Obviously, we have $p = f(\boldsymbol{p})$ and $\hat{p} = f(\widehat{\boldsymbol{p}})$. We also denote that $\xi = \varphi(p)$, $\hat{\xi} = \varphi(\hat{p})$, $\xi_{i,y,z} = \varphi(p_{i,y,z})$ and $\hat{\xi}_{i,y,z} = \varphi(\hat{p}_{i,y,z})$, respectively. Let $\epsilon = |p - \hat{p}| = |f(\boldsymbol{p}) - f(\widehat{\boldsymbol{p}})|$, $\delta = |\varphi(p) - \varphi(\hat{p})| = |\varphi(f(\boldsymbol{p})) - \varphi(f(\widehat{\boldsymbol{p}}))|$, $\epsilon_{i,y,z} = |p_{i,y,z} - \hat{p}_{i,y,z}|$, and $\delta_{i,y,z} = |\varphi(p_{i,y,z}) - \varphi(\hat{p}_{i,y,z})|$ be the respective errors. We have

$$\delta = \left|\xi - \hat{\xi}\right| = |\varphi(f(\boldsymbol{p})) - \varphi(f(\widehat{\boldsymbol{p}}))| = \left|\varphi\left(f\left(\varphi^{-1}\left(\boldsymbol{q}\right)\right)\right) - \varphi\left(f\left(\varphi^{-1}\left(\widehat{\boldsymbol{q}}\right)\right)\right)\right|.$$

Due to the Mean Value Theorem, there exist $\tilde{\xi}_{i,y,z} \in [0,1]$ and accordingly $\tilde{p}_{i,y,z} = \varphi^{-1}(\tilde{\xi}_{i,y,z})$, $1 \le i \le d,\ y, z \in \mathcal{X},\ y \neq z$, such that

$$\delta = \sum_{\substack{1 \le i \le d \\ y \neq z}} \left|\frac{\partial f(\widetilde{\boldsymbol{p}})}{\partial \tilde{p}_{i,y,z}}\right| \frac{\Phi(f(\widetilde{\boldsymbol{p}}))}{\Phi(\tilde{p}_{i,y,z})} \cdot \delta_{i,y,z} \le \kappa(\widetilde{\boldsymbol{p}}) \cdot \max_{\substack{1 \le i \le d \\ y \neq z}} \delta_{i,y,z},$$

where $\kappa(\boldsymbol{p})$ is defined by (7). Since $\mathbb{M}$ satisfies the amortized decay condition, there exists a universal constant $\kappa < 1$ such that $\kappa(\widetilde{\boldsymbol{p}}) \le \kappa$. And since $\mathbb{M}$ is closed under partial pinning, every $\Omega^i_{v,y}$ still belongs to $\mathbb{M}$. Therefore, by induction we have that $\delta \le \kappa^t \delta_0$, where $\delta_0 = |\varphi(p_0) - \varphi(\hat{p}_0)|$ such that $p_0 = \mathbb{P}_{\Omega'}(X_u = w)$ and $\hat{p}_0 = \widehat{\mathbb{P}}^{(0)}_{\Omega'}(X_u = w)$ for some $\Omega' \in \mathbb{M}$, $u \in V_{\Omega'}$ and $x \in \mathcal{X}$, where Ω' is an MRF instance resulting from applying t partial pinnings on the original Ω.

By the Mean Value Theorem, there exists a $\tilde{p}_0 \in [0,1]$ such that $\delta_0 = |\varphi(p_0) - \varphi(\hat{p}_0)| \le |\Phi(\tilde{p}_0)|$, which is upper bounded by q^c for some constant c due to the requirement of amortized decay condition, thus $\delta \le \kappa^t \delta_0 \le q^c \kappa^t$. Recall that $\delta = |\varphi(p) - \varphi(\hat{p})|$. Also by the Mean Value Theorem there exists $\tilde{p} \in [0,1]$ such that $\delta = |\varphi(p) - \varphi(\hat{p})| = |\Phi(\tilde{p})||p - \hat{p}| = |\Phi(\tilde{p})|\epsilon$, thus $\epsilon = \frac{\delta}{|\Phi(\tilde{p})|} \le q^c \delta$. Altogether we have that

$$\left|\mathbb{P}_{\Omega}(X_v = x) - \widehat{\mathbb{P}}^{(t)}_{\Omega}(X_v = x)\right| = \epsilon \le q^c \delta \le q^c \kappa^t \delta_0 \le q^{2c} \kappa^t.$$

And this holds for every $\Omega \in \mathbb{M}, v \in V_{\Omega}, x \in \mathcal{X}$ and $t \ge 1$, with the universal constants c and $\kappa < 1$, which implies the exponential correlation decay of computation tree recursion over $\mathbb{M}$.

The following lemma is proved by verifying the amortized decay condition.

Lemma 2. *Let $\mathbb{M}$ be a family of MRFs satisfying* (1)*. The computation tree recursion exhibits exponential correlation decay over $\mathbb{M}$.*

Proof. Let $\mathbb{M}^*$ be the closure of $\mathbb{M}$ under partial pinning, thus every instance from $\mathbb{M}^*$ is either an instance $\Omega \in \mathbb{M}$ or an outcome of successive partial pinnings of it, and the family $\mathbb{M}^*$ is closed under partial pinning. We show that $\mathbb{M}^*$ satisfies

the amortized decay condition. We choose a monotone function $\varphi : [0,1] \to \mathbb{R}$ so that its derivative Φ satisfies that $\Phi(p) = \left(p + \frac{1}{100q}\right)^{-1}$. Thus both $\Phi(\cdot)$ and $\frac{1}{\Phi(\cdot)}$ are bounded by polynomial of q over $[0,1]$.

Let $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F}) \in \mathbb{M}^*$ be an MRF instance on an underlying graph G with maximum degree Δ, $v \in V_G$ a vertex with $d = \deg(v)$, and $x \in \mathcal{X}$ a spin state. Let $f = f_{\Omega,v,x}$ be the recursion defined by (6).

We define some shorthand notations. For each $1 \le i \le d$ and $y, z \in \mathcal{X}$ that $y \neq z$, let $a_{i,y,z} = 1 - \frac{A_{vv_i}(y,z)}{A_{vv_i}(y,y)}$ and $b_y = F_v(y) \prod_{i=1}^{d} A_{vv_i}(y,y)$, and denote that $s_{i,y} = 1 - \sum_{z \neq y} a_{i,y,z} \cdot p_{i,y,z}$, $s_y = b_y \prod_{i=1}^{d} s_{i,y}$, and $s = \sum_{y \in \mathcal{X}} s_y$. Then we have

$$f(\boldsymbol{p}) = \frac{b_x \prod_{i=1}^{d} \left(1 - \sum_{z \neq x} a_{i,x,z} \cdot p_{i,x,z}\right)}{\sum_{y \in \mathcal{X}} b_y \prod_{i=1}^{d} \left(1 - \sum_{z \neq y} a_{i,y,z} \cdot p_{i,y,z}\right)} = \frac{s_x}{s}.$$

For $\boldsymbol{p} = (p_{i,y,z}, 1 \le i \le d; y, z \in \mathcal{X}; y \neq z) \in [0,1]^{dq(q-1)}$ such that $\sum_{z \neq y} p_{i,y,z} \le 1$ for all i and y, it holds that $s_{i,y} \ge 0$ for any i and y, and $f(\boldsymbol{p}) \in [0,1]$. The partial derivatives satisfy:

$$\left|\frac{\partial f(\boldsymbol{p})}{\partial p_{i,x,z}}\right| = \left|\frac{a_{i,x,z} s_x (s - s_x)}{s^2 \cdot s_{i,x}}\right| = f(\boldsymbol{p})(1 - f(\boldsymbol{p}))\frac{|a_{i,x,z}|}{s_{i,x}},$$

$$\sum_{y \neq x} \left|\frac{\partial f(\boldsymbol{p})}{\partial p_{i,y,z}}\right| = \sum_{y \neq x} \left|\frac{a_{i,y,z} s_x s_y}{s^2 \cdot s_{i,y}}\right| = f(\boldsymbol{p}) \sum_{y \neq x} \frac{s_y}{s} \sum_{i=1}^{d} \frac{|a_{i,y,z}|}{s_{i,y}}.$$

The amortized decay rate defined by (7) is then bounded as

$$\begin{aligned}
\kappa(\boldsymbol{p}) &= \sum_{\substack{1 \le i \le d \\ y \neq z}} \left|\frac{\partial f(\boldsymbol{p})}{\partial p_{i,y,z}}\right| \frac{\Phi(f(\boldsymbol{p}))}{\Phi(p_{i,y,z})} \\
&= f(\boldsymbol{p})(1 - f(\boldsymbol{p}))\Phi(f(\boldsymbol{p})) \sum_{i=1}^{d} \frac{1}{s_{i,x}} \sum_{z \neq x} \frac{|a_{i,x,z}|}{\Phi(p_{i,x,z})} \\
&\quad + f(\boldsymbol{p})\Phi(f(\boldsymbol{p})) \sum_{y \neq x} \frac{s_y}{s} \sum_{i=1}^{d} \frac{1}{s_{i,y}} \sum_{z \neq y} \frac{|a_{i,y,z}|}{\Phi(p_{i,y,z})} \\
&\le \sum_{\substack{1 \le i \le d \\ z \neq x}} \frac{|a_{i,x,z}|}{s_{i,x}} \cdot \left(p_{i,y,z} + \frac{1}{100q}\right) + \max_{y \neq x} \sum_{\substack{1 \le i \le d \\ z \neq y}} \frac{|a_{i,y,z}|}{s_{i,y}} \cdot \left(p_{i,y,z} + \frac{1}{100q}\right) \\
&\le \frac{101}{50}\Delta \cdot \max_{\substack{1 \le i \le d \\ z \neq y}} \frac{|a_{i,y,z}|}{s_{i,y}}.
\end{aligned}$$

For every $1 \le i \le d$ and $y, z \in \mathcal{X}$ that $y \neq z$, it can be verified that $s_{i,y} = p_{i,y,y} + \sum_{z \neq y} \frac{A_{vv_i}(y,z)}{A_{vv_i}(y,y)} \cdot p_{i,y,z} \ge \frac{1}{c_{\boldsymbol{A}}}$, and

$$|a_{i,y,z}| = \left|1 - \frac{A_{vv_i}(y,z)}{A_{vv_i}(y,y)}\right| \le \max\left\{\frac{c_{\boldsymbol{A}} - 1}{c_{\boldsymbol{A}}}, c_{\boldsymbol{A}} - 1\right\} \le c_{\boldsymbol{A}} - 1.$$

Note that the partial pinning does not affect the edge activity $\boldsymbol{A}$, thus for $\mathbb{M}$ satisfying (1), for every $\Omega \in \mathbb{M}^*$ the $c_{\boldsymbol{A}}$ still satisfies that $3\Delta(c_{\boldsymbol{A}} - 1) \leq 1$. Therefore,

$$\kappa(\boldsymbol{p}) \leq \frac{101}{50}\Delta c_{\boldsymbol{A}}(c_{\boldsymbol{A}} - 1) \leq \frac{101}{150}(1 + \frac{1}{3\Delta}) \leq \frac{404}{450} < 1.$$

Therefore, the MRF family $\mathbb{M}^*$ satisfies the amortized decay condition. By Lemma 1, the computation tree recursion exhibits exponential correlation decay over $\mathbb{M}^*$ thus also over its subfamily $\mathbb{M}$.

Proof of Theorem 3: Let $\Omega = (G, \mathcal{X}, \boldsymbol{A}, \boldsymbol{F}) \in \mathbb{M}$ be an MRF instance and $G = (V, E)$. Enumerate the vertices in V as $v_1, v_2, \ldots, v_n$. For each $1 \leq k \leq n$, let $\mathbb{P}^{(t)}_{\Omega_k}(X_{v_k} = x_k)$ be computed by the algorithm in Section 3.1, where $\Omega_1 = \Omega$ and $\Omega_{k+1} = \textsc{Pin}_{v_k, x_k}(\Omega_k)$. It is easy to verify that Ω_k still satisfies the condition (1) for every k since pinning increases neither Δ nor $c_{\boldsymbol{A}}$. Let $\widehat{Z}(\Omega) = \frac{w(\boldsymbol{x})}{\prod_{k=1}^{n} \widehat{\mathbb{P}}^{(t)}_{\Omega_k}(X_{v_k} = x_{v_k})}$ where $w(\boldsymbol{x}) = \prod_{e=uv \in E} A_e(x_u, x_v) \prod_{v \in V} F_v(x_v)$. It holds that

$$\mathbb{P}_{\Omega}(\boldsymbol{X} = \boldsymbol{x}) = \prod_{k=1}^{n} \mathbb{P}_{\Omega}(X_{v_k} = x_k \mid \forall 1 \leq i < k, X_{v_i} = x_i).$$

As observed in [10], the marginal probability $\mathbb{P}_{\Omega}(X_{v_k} = x_k \mid \forall 1 \leq i < k, X_{v_i} = x_i) = \mathbb{P}_{\Omega_k}(X_{v_k} = x_k)$.

Since Ω_k satisfies the condition (1), by Lemma 2, there exists constant $C > 0$ such that

$$\left|\mathbb{P}_{\Omega_k}(X_{v_k} = x_{v_k}) - \widehat{\mathbb{P}}^{(t)}_{\Omega_k}(X_{v_k} = x_{v_k})\right| \leq \mathrm{poly}(|\Omega|) \cdot \exp(-C \cdot t).$$

Thus by choosing appropriate $t = O\left(\log \frac{1}{\epsilon} + \log q + \log n\right)$, it holds for every k that

$$\left|\mathbb{P}_{\Omega_k}(X_{v_k} = x_{v_k}) - \widehat{\mathbb{P}}^{(t)}_{\Omega_k}(X_{v_k} = x_{v_k})\right| \leq \frac{\epsilon}{4qn},$$

and since in the algorithm we always choose the x_{v_k} maximizing the value of $\widehat{\mathbb{P}}^{(t)}_{\Omega_k}(X_{v_k} = x_{v_k})$, we have $\widehat{\mathbb{P}}^{(t)}_{\Omega_k}(X_{v_k} = x_{v_k}) \geq \frac{1}{q}$ thus $\mathbb{P}_{\Omega_k}(X_{v_k} = x_{v_k}) \geq \frac{1}{q} - \frac{\epsilon}{4qn} \geq \frac{1}{2q}$.

By definition we have $\mathbb{P}_{\Omega}(\boldsymbol{X} = \boldsymbol{x}) = \frac{w(\boldsymbol{x})}{Z(\Omega)}$, thus $Z(\Omega) = \frac{w(\boldsymbol{x})}{\prod_{k=1}^{n} \mathbb{P}_{\Omega_k}(X_{v_k} = x_{v_k})}$. Therefore, we have

$$1 - \epsilon \leq \left(1 - \frac{\epsilon}{2n}\right)^n \leq \frac{Z(\Omega)}{\widehat{Z}(\Omega)} = \prod_{k=1}^{n} \frac{\widehat{\mathbb{P}}^{(t)}_{\Omega_k}(X_{v_k} = x_{v_k})}{\mathbb{P}_{\Omega_k}(X_{v_k} = x_{v_k})} \leq \left(1 + \frac{\epsilon}{2n}\right)^n \leq 1 + \epsilon,$$

which is simplified as that $1 - \epsilon \leq \frac{\widehat{Z}(\Omega)}{Z(\Omega)} \leq 1 + \epsilon$.

By Proposition 2, the total running time is bounded by $\mathrm{poly}(|\Omega|)(q\Delta)^{O(t)}$. Since $t = O\left(\log \frac{1}{\epsilon} + \log q + \log n\right)$, the algorithm is an FPTAS if q and Δ are constants, and in general the running time is bounded by $|\Omega|^{O(\log |\Omega|)}$ for any fixed $0 < \epsilon < 1$. □

4 List-coloring

We consider list-coloring instances $\Omega = (G, \mathcal{X}, \boldsymbol{L})$ satisfying the condition (3). Let $\Delta = \Delta_G$ be the maximum degree of G and define that $\chi(\Delta) = (\alpha - 1)\Delta + 1$. The condition (3) implies the following weaker condition:

$$\forall v \in V, \quad |L_v| - \deg(v) \geq \chi(\Delta). \tag{8}$$

A merit of considering this weaker condition is that it is closed under partial pinning and pinning. The pinning and partial pinning can be defined on list-coloring instances as they are special cases of MRFs. Given a list-coloring instance $\Omega = (G, \mathcal{X}, \boldsymbol{L})$ with underlying graph $G = (V, E)$ and a vertex $v \in V$ with d neighbors $v_1, v_2, \ldots, v_d$, for each color $x \in L_v$, the pinning of Ω is a new list-coloring instance $\Omega_{v,x} = \textsc{Pin}_{v,x}(\Omega) = (G_v, \mathcal{X}, \widehat{\boldsymbol{L}})$, where G_v is the subgraph of G induced by $V \setminus \{v\}$ and $\widehat{\boldsymbol{L}} = (\widehat{L}_u, u \in V \setminus \{v\})$ such that $\widehat{L}_u = L_u \setminus \{x\}$ if u is adjacent to v and $\widehat{L}_u = L_u$ if otherwise; and for each $1 \leq i \leq d+1$, the partial pinning of Ω is a new list-coloring instance $\Omega^i_{v,x} = (G_v, \mathcal{X}, \widetilde{\boldsymbol{L}})$, where $\widetilde{\boldsymbol{L}} = (\widetilde{L}_u, u \in V \setminus \{v\})$ such that $\widetilde{L}_u = L_u \setminus \{x\}$ for $u = v_j$ with $j < i$ and $\widetilde{L}_u = L_u$ for all other u in $V \setminus \{v\}$. The pinning and the partial pinning does not violate the condition (8) since it never increases the maximum degree, and if $|L_v|$ decreases by 1 then also $\deg(v)$ decreases by 1.

The following identity for marginals of list-coloring is proved in [10].

Proposition 3. *Let $\Omega = (G, \mathcal{X}, \boldsymbol{L})$ be a list-coloring instance on graph $G = (V, E)$, $v \in V$ a vertex with d neighbors $v_1, v_2, \ldots, v_d$ where $d = \deg(v)$, and $x \in L_v$ a color. It holds that*

$$\mathbb{P}_\Omega(X_v = x) = \frac{\prod_{i=1}^{d} \left(1 - \mathbb{P}_{\Omega^i_{v,x}}(X_{v_i} = x)\right)}{\sum_{y \in L_v} \prod_{i=1}^{d} \left(1 - \mathbb{P}_{\Omega^i_{v,x}}(X_{v_i} = y)\right)}.$$

Some simple lower and upper bounds hold for the marginals, similar to the ones proved in [10].

Lemma 3. *Let $\Omega = (G, \mathcal{X}, \boldsymbol{L})$ be a list-coloring instance with the maximum degree Δ of G, satisfying the condition (8). For any vertex $v \in V_G$ and any color $x \in L_v$, it holds for the marginal probability that $\frac{1}{q \cdot \mathrm{e}^{\frac{1}{\alpha-1}}} \leq \mathbb{P}_\Omega(X_v = x) \leq \frac{1}{\chi(\Delta)}$.*

Proof. The upper bound is easy: conditioning on any coloring of the neighbos of v, the number of remaining colors for v is at least $|L_v| - \deg(v) \geq \chi(\Delta)$, thus marginal probability is at most $\frac{1}{\chi(\Delta)}$. Applying the upper bound $\frac{1}{\chi(\Delta)}$ to the marginals in the numerator of the recursion in Proposition 3 and the trivial upper bound q to the denominator, we have the lower bound $\frac{1}{q \cdot \mathrm{e}^{\frac{1}{\alpha-1}}}$.

4.1 The Computation Tree Recursion with Adjustment

Given a list-coloring instance $\Omega = (G, \mathcal{X}, \boldsymbol{L})$ on graph $G = (V, E)$, a vertex $v \in V$ with d neighbors $v_1, v_2, \ldots, v_d$ and a color $x \in L_v$, the computation tree

recursion $f_{\Omega,v,x}$ can be defined on the domain of all $\boldsymbol{p} = (p_{i,y}, 1 \le i \le d \wedge y \in L_v) \in [0,1]^{d|L_v|}$:

$$f_{\Omega,v,x}(\boldsymbol{p}) \triangleq \frac{\prod_{i=1}^{d}(1-p_{i,x})}{\sum_{y\in L_v}\prod_{i=1}^{d}(1-p_{i,y})}. \tag{9}$$

For $t \ge 0$, the quantity $\widehat{\mathbb{P}}_{\Omega}^{(t)}(X_v = x)$ is recursively defined as follows:

- If $t = 0$, let $\widehat{\mathbb{P}}_{\Omega}^{(0)}(X_v = x) = \frac{1}{|L_v|}$.
- If $t > 0$, let $\widehat{\mathbb{P}}_{\Omega}^{(t)}(X_v = x) = \min\left\{\frac{1}{|L_v|-d}, f_{\Omega,v,x}(\widehat{\boldsymbol{p}})\right\}$, where the $\widehat{\boldsymbol{p}}$ is taken as that $\hat{p}_{i,y} = \widehat{\mathbb{P}}_{\Omega^i_{v,x}}^{(t-1)}(X_{v_i} = y)$ for each $1 \le i \le d$ and $y \in L_v$.

Note that the only difference from the MRF case is the truncation of the value of $f(\widehat{\boldsymbol{p}})$ so that $\mathbb{P}_{\Omega}(X_v = x)$ never goes beyond the naive upper bound $\frac{1}{|L_v|-d}$. We call this procedure the *computation tree recursion with adjustment*. It is the same as the procedure proposed in [10] except with a more simplified value truncation.

The estimation $\widehat{Z}(\Omega)$ of the partition function is computed from these estimations $\widehat{\mathbb{P}}_{\Omega}^{(t)}(X_v = x)$ of marginal probabilities by the same algorithm as in Section 3.1. The same complexity bound as in Proposition 2 still holds.

4.2 Correlation Decay

The correlation decay of the computation tree recursion with adjustment can be defined in the same way as in Definition 2.

Lemma 4. *The computation tree recursion with adjustment exhibits exponential correlation decay on list-coloring instances satisfying the condition* (8) *with* $\alpha > \alpha^*$ *where* $\alpha^* \approx 2.58071$ *is defined by* (2) *in Section 2.*

Proof. Let $\Omega = (G, \mathcal{X}, \boldsymbol{L})$ be a list-coloring instance on the underlying graph $G = (V, E)$ with the maximum degree $\Delta = \Delta(G)$ satisfying the condition (8). It can be verified that all the list-coloring instances generated by recursively applying partial pinnings on Ω still satisfy the condition (8).

Let $v \in V$ be a vertex with d neighbors $v_1, v_2, \ldots, v_d$, $x \in L_v$ a color, and $f = f_{\Omega,v,x}$ the recursion defined by (9). It holds that $\mathbb{P}_{\Omega}(X_v = x) = f(\boldsymbol{p})$ where $\boldsymbol{p} = (p_{i,y}, 1 \le i \le d \wedge y \in L_v)$ and each $p_{i,y} = \mathbb{P}_{\Omega^i_{v,x}(X_{v_i}=y)}$. We choose the monotone differentiable function $\varphi : [0,1] \to \mathbb{R}$ so that its derivative is $\Phi(p) = \frac{\mathrm{d}\,\varphi(p)}{\mathrm{d}\,p} = \frac{1}{(1-p)\sqrt{p}}$. We define the amortized decay rate in the same way as (7) by:

$$\kappa(\boldsymbol{p}) \triangleq \sum_{\substack{1\le i\le d\\ y\in L_v}} \left|\frac{\partial f(\boldsymbol{p})}{\partial p_{i,y}}\right| \frac{\Phi(f(\boldsymbol{p}))}{\Phi(p_{i,y})}.$$

By the same analysis as in Lemma 1, due to the mean value theorem, we have

$$\begin{aligned}&\left|\varphi\left(\mathbb{P}_{\Omega}(X_v=x)\right)-\varphi\left(\widehat{\mathbb{P}}_{\Omega}^{(t)}(X_v=x)\right)\right|\\ \le&\kappa(\boldsymbol{p})\cdot\max_{\substack{1\le i\le d\\ y\in L_v}}\left|\varphi\left(\mathbb{P}_{\Omega^i_{v,x}}(X_{v_i}=y)\right)-\varphi\left(\widehat{\mathbb{P}}^{(t-1)}_{\Omega^i_{v,x}}(X_{v_i}=y)\right)\right|,\end{aligned}$$

for some $\boldsymbol{p}=(p_{i,y}, 1\le i\le d\wedge y\in L_v)$ such that the value of each $p_{i,y}$ is between $\mathbb{P}_{\Omega^i_{v,x}}(X_{v_i}=y)$ and $\widehat{\mathbb{P}}^{(t-1)}_{\Omega^i_{v,x}}(X_{v_i}=y)$. By Lemma 3, we have $\mathbb{P}_{\Omega}(X_{v_i}=y)\le\frac{1}{\chi(\Delta)}$ and due to the definition of the algorithm, $\widehat{\mathbb{P}}^{(t)}_{\Omega}(X_{v_i}=y)\le\frac{1}{|L_v|-d}\le\frac{1}{\chi(\Delta)}$, thus $p_{i,y}\le\frac{1}{\chi(\Delta)}$ for any $1\le i\le d$ and $y\in L_v$.

By our choice of $\Phi(\cdot)$, it can be verified that

$$\begin{aligned}\kappa(\boldsymbol{p})&=\sum_{i=1}^{d}\left|\frac{\partial f(\boldsymbol{p})}{\partial p_{i,x}}\right|\frac{\Phi(f(\boldsymbol{p}))}{\Phi(p_{i,x})}+\sum_{\substack{1\le i\le d\\ y\in L_v\setminus\{x\}}}\left|\frac{\partial f(\boldsymbol{p})}{\partial p_{i,y}}\right|\frac{\Phi(f(\boldsymbol{p}))}{\Phi(p_{i,y})}\\ &\le\sqrt{f(\boldsymbol{p})}\left(\sum_{i=1}^{d}\sqrt{p_{i,x}}+\sum_{i=1}^{d}\max_{y\ne x}\sqrt{p_{i,y}}\right)\\ &\le\sqrt{\frac{\prod_{i=1}^{d}(1-p_{i,x})}{(d+\chi(\Delta))\left(1-\frac{1}{\chi(\Delta)}\right)^d}}\left(\sum_{i=1}^{d}\sqrt{p_{i,x}}+\frac{d}{\sqrt{\chi(\Delta)}}\right),\end{aligned}\tag{10}$$

where the last inequality is due to that $p_{i,x}\le\frac{1}{\chi(\Delta)}$ and $|L_v|\le d+\chi(\Delta)$. Let $\bar{p}=1-\left(\prod_{i=1}^{d}(1-p_{i,x})\right)^{\frac{1}{d}}$. Then $\bar{p}\le\frac{1}{\chi(\Delta)}$ since all $p_{i,x}$ satisfy so, and $\prod_{i=1}^{d}(1-p_{i,x})=(1-\bar{p})^d$. Let $\ell_i=\ln(1-p_{i,x})$, thus $\sum_{i=1}^{d}\ell_i=d\ln(1-\bar{p})$. The function $g(x)=\sqrt{1-\mathrm{e}^x}$ is concave over $x\le 0$, thus by Jensen's inequality,

$$\sum_{i=1}^{d}\sqrt{p_{i,x}}=\sum_{i=1}^{d}g(\ell_i)\le d\cdot g\left(\frac{1}{d}\sum_{i=1}^{d}\ell_i\right)=d\sqrt{\bar{p}}.$$

Therefore, (10) can be bounded by its symmetrized form as follows:

$$\begin{aligned}\kappa(\boldsymbol{p})\le\kappa(\bar{p})&\triangleq\frac{d}{\sqrt{d+\chi(\Delta)}}\left(\frac{1-\bar{p}}{1-\frac{1}{\chi(\Delta)}}\right)^{\frac{d}{2}}\left(\sqrt{\bar{p}}+\frac{1}{\sqrt{\chi(\Delta)}}\right)\\ &\le\frac{\Delta}{\sqrt{\Delta+\chi(\Delta)}}\left(\frac{1-\bar{p}}{1-\frac{1}{\chi(\Delta)}}\right)^{\frac{\Delta}{2}}\left(\sqrt{\bar{p}}+\frac{1}{\sqrt{\chi(\Delta)}}\right).\end{aligned}$$

where the last inequality is due to that $\bar{p}\le\frac{1}{\chi(\Delta)}$ and $d\le\Delta$.

Let $\bar{p}=\frac{\rho}{\chi(\Delta)}$ for $\rho\in[0,1]$. It holds that $\kappa(\bar{p})\le\frac{(\sqrt{\rho}+1)}{\sqrt{\alpha(\alpha-1)}}\exp\left(-\frac{\rho-1}{2(\alpha-1)}\right)$, whose maximum is achieved when $\rho=\frac{1}{2}(2\alpha-1-\sqrt{4\alpha-3})$, such that

$$\kappa(\bar{p})\le\kappa_\alpha\triangleq\frac{\sqrt{2}+\sqrt{2\alpha-1-\sqrt{4\alpha-3}}}{\sqrt{2\alpha(\alpha-1)}}\exp\left(\frac{3-2\alpha+\sqrt{4\alpha-3}}{4(\alpha-1)}\right).$$

It can be verified that κ_α is is monotonously decreasing from $+\infty$ to 0 for $\alpha > 1$, so $\kappa_\alpha < 1$ if $\alpha > \alpha^*$ where α^* is the unique solution to $\kappa_\alpha = 1$, as defined by (2).

Since the condition (8) is closed under partial pinning, by induction we have

$$\begin{aligned}&\left|\varphi\left(\mathbb{P}_{\Omega}(X_v = x)\right) - \varphi\left(\widehat{\mathbb{P}}^{(t)}_{\Omega}(X_v = x)\right)\right| \\ \leq &\kappa^t \left|\varphi\left(\mathbb{P}_{\Omega'}(X_u = z)\right) - \varphi\left(\widehat{\mathbb{P}}^{(0)}_{\Omega'}(X_u = z)\right)\right|,\end{aligned}$$

where $\Omega' = (G', \mathcal{X}, \boldsymbol{L}')$ is a list-coloring instance resulting from recursively applying t partial pinnings on the original Ω. By the same mean value theorem argument as in Lemma 1, we have $\left|\mathbb{P}_{\Omega}(X_v = x) - \widehat{\mathbb{P}}^{(t)}_{\Omega}(X_v = x)\right| \leq \frac{\Phi(\tilde{p}_0)}{\Phi(\tilde{p})}\kappa_\alpha^t$, for some $\tilde{p} \in [0, 1]$ and some $\tilde{p}_0$ between $\mathbb{P}_{\Omega'}(X_u = w)$ and $\widehat{\mathbb{P}}^{(0)}_{\Omega'}(X_u = w) = \frac{1}{|L'_u|}$. Recall that the condition (8) is closed under partial pinning. It holds that $\frac{1}{q} \leq \frac{1}{|L'_v|} \leq \frac{1}{\chi(\Delta(G'))}$, and by Lemma 3 it also holds that $\frac{1}{q \cdot \mathrm{e}^{1/(\alpha-1)}} \leq \mathbb{P}_{\Omega'}(X_u = w) \leq \frac{1}{\chi(\Delta(G'))}$. Therefore, $\tilde{p}_0 \in \left[\frac{1}{q \cdot \mathrm{e}^{1/(\alpha-1)}}, \frac{1}{\chi(\Delta(G'))}\right]$. By our choice of $\Phi(p)$, we have $\frac{\Phi(\tilde{p}_0)}{\Phi(\tilde{p})} \leq \frac{\sqrt{q} \cdot \mathrm{e}^{\frac{1}{2(\alpha-1)}}}{1 - \frac{1}{\chi(\Delta(G'))}} = O(\sqrt{q})$.

In conclusion, if the condition (8) is satisfied with $\alpha > \alpha^* \approx 2.58071$, there exists a constant $\kappa < 1$ such that $\left|\mathbb{P}_{\Omega}(X_v = x) - \widehat{\mathbb{P}}^{(t)}_{\Omega}(X_v = x)\right| \leq O(\sqrt{q})\kappa^t$.

Proof of Theorem 4: We first prove the theorem under the weaker condition (8), which is closed under pinning and partial pinning. The proof is the same as the proof of Theorem 3. The theorem with the stronger condition (3) follows as a consequence. □

References

1. Bandyopadhyay, A., Gamarnik, D.: Counting without sampling: Asymptotics of the log-partition function for certain statistical physics models. Random Structures & Algorithms 33(4), 452–479 (2008)
2. Bayati, M., Gamarnik, D., Katz, D., Nair, C., Tetali, P.: Simple deterministic approximation algorithms for counting matchings. In: Proceedings of STOC, pp. 122–127 (2007)
3. Cai, J.-Y., Chen, X.: A decidable dichotomy theorem on directed graph homomorphisms with non-negative weights. In: Proceedings FOCS, pp. 437–446 (2010)
4. Cai, J.-Y., Chen, X., Lu, P.: Graph homomorphisms with complex values: A dichotomy theorem. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 275–286. Springer, Heidelberg (2010)
5. Dyer, M., Jerrum, M., Vigoda, E.: Rapidly mixing markov chains for dismantleable constraint graphs. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 68–77. Springer, Heidelberg (2002)
6. Dyer, M.E., Frieze, A.M., Hayes, T.P., Vigoda, E.: Randomly coloring constant degree graphs. In: Proceedings of FOCS, pp. 582–589 (2004)
7. Dyer, M.E., Greenhill, C.S.: On markov chains for independent sets. Journal of Algorithms 35(1), 17–49 (2000)

8. Galanis, A., Štefankovič, D., Vigoda, E.: Inapproximability of the partition function for the anti-ferromagnetic ising and hard-core models. arXiv preprint arXiv:1203.2226 (2012)
9. Galanis, A., Štefankovič, D., Vigoda, E.: Inapproximability for anti-ferromagnetic spin systems in the tree non-uniqueness region. arXiv preprint arXiv:1305.2902 (2013)
10. Gamarnik, D., Katz, D.: Correlation decay and deterministic FPTAS for counting colorings of a graph. Journal of Discrete Algorithms 12, 29–47 (2012)
11. Gamarnik, D., Katz, D., Misra, S.: Strong spatial mixing for list coloring of graphs. arXiv preprint arXiv:1207.1223 (2012)
12. Goldberg, L.A., Martin, R., Paterson, M.: Strong spatial mixing with fewer colors for lattice graphs. SIAM Journal on Computing 35(2), 486 (2005)
13. Goldberg, L.A., Jerrum, M.: A polynomial-time algorithm for estimating the partition function of the ferromagnetic ising model on a regular matroid. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 521–532. Springer, Heidelberg (2011)
14. Goldberg, L.A., Jerrum, M., Paterson, M.: The computational complexity of two-state spin systems. Random Structures & Algorithms 23(2), 133–154 (2003)
15. Hayes, T.P.: Randomly coloring graphs of girth at least five. In: Proceedings of STOC, pp. 269–278 (2003)
16. Jerrum, M.: A very simple algorithm for estimating the number of k-colorings of a low-degree graph. Random Structures & Algorithms 7(2), 157–166 (1995)
17. Jerrum, M., Sinclair, A.: Polynomial-time approximation algorithms for the ising model. SIAM Journal on Computing 22(5), 1087–1116 (1993)
18. Jerrum, M., Sinclair, A., Vigoda, E.: A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. Journal of the ACM 51, 671–697 (2004)
19. Li, L., Lu, P., Yin, Y.: Approximate counting via correlation decay in spin systems. In: Proceedings of SODA, pp. 922–940 (2012)
20. Li, L., Lu, P., Yin, Y.: Correlation decay up to uniqueness in spin systems. In: Proceedings of SODA, pp. 67–84 (2013)
21. Luby, M., Vigoda, E.: Approximately counting up to four (extended abstract). In: Proceedings of STOC, pp. 682–687 (1997)
22. Restrepo, R., Shin, J., Tetali, P., Vigoda, E., Yang, L.: Improved mixing condition on the grid for counting and sampling independent sets. In: Proceedings of FOCS, pp. 140–149 (2011)
23. Sinclair, A., Srivastava, P., Thurley, M.: Approximation algorithms for two-state anti-ferromagnetic spin systems on bounded degree graphs. In: Proceedings of SODA, pp. 941–953 (2012)
24. Sly, A.: Uniqueness thresholds on trees versus graphs. The Annals of Applied Probability 18(5), 1897–1909 (2008)
25. Sly, A.: Computational transition at the uniqueness threshold. In: Proceedings of FOCS, pp. 287–296 (2010)
26. Sly, A., Sun, N.: The computational hardness of counting in two-spin models on d-regular graphs. In: Proceedings of FOCS, pp. 361–369 (2012)
27. Vigoda, E.: Improved bounds for sampling coloring. In: Proceedings of FOCS, pp. 51–59 (1999)
28. Weitz, D.: Counting independent sets up to the tree threshold. In: Proceedings of STOC, pp. 140–149 (2006)
29. Yin, Y., Zhang, C.: Approximate counting via correlation decay on planar graphs. In: Proceedings of SODA, pp. 47–66 (2013)

Pseudorandomness for Regular Branching Programs via Fourier Analysis

Omer Reingold[1], Thomas Steinke[2,⋆], and Salil Vadhan[2,⋆⋆]

[1] Microsoft Research Silicon Valley, Mountain View, CA
Omer.Reingold@microsoft.com
[2] School of Engineering and Applied Sciences, Harvard University, Cambridge MA
{tsteinke,salil}@seas.harvard.edu

Abstract. We present an explicit pseudorandom generator for oblivious, read-once, permutation branching programs of constant width that can read their input bits in any order. The seed length is $O(\log^2 n)$, where n is the length of the branching program. The previous best seed length known for this model was $n^{1/2+o(1)}$, which follows as a special case of a generator due to Impagliazzo, Meka, and Zuckerman (FOCS 2012) (which gives a seed length of $s^{1/2+o(1)}$ for arbitrary branching programs of size s). Our techniques also give seed length $n^{1/2+o(1)}$ for general oblivious, read-once branching programs of width $2^{n^{o(1)}}$, which is incomparable to the results of Impagliazzo et al.

Our pseudorandom generator is similar to the one used by Gopalan et al. (FOCS 2012) for read-once CNFs, but the analysis is quite different; ours is based on Fourier analysis of branching programs. In particular, we show that an oblivious, read-once, *regular* branching program of width w has Fourier mass at most $(2w^2)^k$ at level k, independent of the length of the program.

1 Introduction

A major open problem in the theory of pseudorandomness is to construct an "optimal" pseudorandom generator for space-bounded computation. That is, we want an explicit pseudorandom generator that stretches a uniformly random seed of length $O(\log n)$ to n bits that cannot be distinguished from uniform by any $O(\log n)$-space algorithm (which receives the pseudorandom bits one at a time, in a streaming fashion, and may be nonuniform).

Such a generator would imply that every randomized algorithm can be derandomized with only a constant-factor increase in space (RL = L), and would also have a variety of other applications, such as in streaming algorithms [1], deterministic dimension reduction and SDP rounding [2], hashing [3], hardness

⋆ Work done in part while at Stanford University. Supported by NSF grant CCF-1116616 and the Lord Rutherford Memorial Research Fellowship.

⋆⋆ Work done in part when on leave as a Visiting Researcher at Microsoft Research Silicon Valley and a Visiting Scholar at Stanford University. Supported in part by NSF grant CCF-1116616 and US-Israel BSF grant 2010196.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 655–670, 2013.

amplification [4], almost k-wise independent permutations [5], and cryptographic pseudorandom generator constructions [6].

Unfortunately, for fooling general logspace algorithms, there has been essentially no improvement since the classic work of Nisan [7], which provided a pseudorandom generator of seed length $O(\log^2 n)$. Instead, a variety of works have improved the seed length for various restricted classes of logspace algorithms, such as algorithms that use $n^{o(1)}$ random bits [8, 9], combinatorial rectangles [10–13] random walks on graphs [14, 15], branching programs of width 2 or 3 [16–18], and regular or permutation branching programs (of bounded width) [19–23].

The vast majority of these works are based on Nisan's generator or its variants by Impagliazzo, Nisan, and Wigderson [24] and Nisan and Zuckerman [8], and show how the analysis (and hence the final parameters) of these generators can be improved for logspace algorithms that satisfy the additional restrictions. All three of these generators are based on recursive use of the following principle: if we consider two consecutive time intervals I_1, I_2 in a space s computation and use some randomness r to generate the pseudorandom bits fed to the algorithm during interval I_1, then at the start of I_2, the algorithm will 'remember' at most s bits of information about r. So we can use a randomness extractor to extract roughly $|r|-s$ almost uniform bits from r (while investing only a small additional amount of randomness for the extraction). This paradigm seems unlikely to yield pseudorandom generators for general logspace computations that have a seed length of $\log^{1.99} n$ (see [20]).

Thus, there is a real need for a different approach to constructing pseudorandom generators for space-bounded computation. One new approach has been suggested in the recent work of Gopalan et al. [25], which constructed improved pseudorandom generators for read-once CNF formulas and combinatorial rectangles, and hitting set generators for width 3 branching programs. Their basic generator (e.g. for read-once CNF formulas) works as follows: Instead of considering a fixed partition of the bits into intervals, they pseudorandomly partition the bits into two groups, assign the bits in one group using a small-bias generator [26], and then recursively generate bits for the second group. While it would not work to assign *all* the bits using a single sample from a small-bias generator, it turns out that generating a pseudorandom partial assignment is a significantly easier task.

An added feature of the Gopalan et al. generator is that its pseudorandomness properties are independent of the order in which the output bits are read by a potential distinguisher. In contrast, Nisan's generator and its variants depend heavily on the ordering of bits (the intervals I_1 and I_2 above cannot be interleaved), and in fact it is known that a particular instantiation of Nisan's generator fails to be pseudorandom if the (space-bounded) distinguisher can read the bits in a different order [27, Corollary 3.18]. Recent works [28, 29] have constructed nontrivial pseudorandom generators for space-bounded algorithms that can read their bits in any order, but the seed length achieved is larger than $\sqrt{n}$.

In light of the above, a natural question is whether the approach of Gopalan et al. can be extended to a wider class of space-bounded algorithms. We make progress on this question by using the same approach to construct a pseudorandom generator with seed length $O(\log^2 n)$ for constant-width, read-once, oblivious permutation branching programs that can read their bits in any order. In analysing our generator, we develop new Fourier-analytic tools for proving pseudorandomness against space-bounded algorithms.

1.1 Models of Space-Bounded Computation

A (layered) **branching program** B is a nonuniform model of space-bounded computation. The program maintains a **state** from the set $[w] = \{1, \ldots, w\}$ and, at each time step i, reads one bit of its input $x \in \{0,1\}^n$ and updates its state according to a transition function $B_i : \{0,1\} \times [w] \to [w]$. The parameter w is called the **width** of the program, and corresponds to a space bound of $\log w$ bits. We allow the transition function B_i to be different at each time step i. We consider several restricted forms of branching programs:

- **Read-once branching programs** read each input bit at most once.
- **Oblivious branching programs** choose which input bit to read depending only on the time step i, and not on the current state
- **Ordered branching programs** (a.k.a. streaming algorithms) always read input bit i in time step i (hence are necessarily both read-once and oblivious).

To derandomize randomized space-bounded computations (e.g. prove RL = L), it suffices to construct pseudorandom generators that fool ordered branching programs of polynomial width ($w = \mathrm{poly}(n)$), and hence this is the model addressed by most previous constructions (including Nisan's generator). However, the more general models of oblivious and read-once branching programs are also natural to study, and, as discussed above, can spark the development of new techniques for reasoning about pseudorandomness.

As mentioned earlier, Nisan's pseudorandom generator [7] achieves $O(\log^2 n)$ seed length for *ordered* branching programs of polynomial width. It is known how to achieve $O(\log n)$ seed length for ordered branching programs width 2 [17], and for width 3, it is only known how to construct "hitting-set generators" (a weaker form of pseudorandom generators) with seed length $O(\log n)$ [18, 25]. (The seed length is $\tilde{O}(\log n)$ if we want the error of the hitting set generator to be subconstant.) For pseudorandom generators for width $w \geq 3$ and hitting-set generators for width $w \geq 4$, there is no known construction with seed length $o(\log^2 n)$.

The study of pseudorandomness against non-ordered branching programs started more recently. Tzur [27] showed that there are oblivious, read-once, constant-width branching programs that can distinguish the output of Nisan's generator from uniform. Bogdanov, Papakonstantinou, and Wan [28] exhibited a pseudorandom generator with seed length $(1 - \Omega(1)) \cdot n$ for oblivious read-once branching programs of width w for $w = 2^{\Omega(n)}$. Impagliazzo, Meka, and Zuckerman [29] gave a pseudorandom generator with seed length $s^{1/2+o(1)}$ for arbitrary

branching programs of size s; note that $s = O(nw)$ for a read-once branching program of width w and length n.

We consider two further restrictions on branching programs:

- **Regular branching programs** are oblivious branching programs with the property that, if the distribution on states in any layer is uniformly random and the input bit read by the program at that layer is uniformly random, then the resulting distribution on states in the next layer is uniformly random. This is equivalent to requiring that the bipartite graph associated with each layer of the program, where we have edges from each state $u \in [w]$ in layer i to the possible next-states $u_0, u_1 \in [w]$ in layer $i + 1$ (if the input bit is b, the state goes to u_b), is a regular graph.
- **Permutation branching programs** are a further restriction, where we require that for each setting of the input string, the mappings between layers are permutations. This is equivalent to saying that (regular) bipartite graphs corresponding to each layer are decomposed into two perfect matchings, one corresponding to each value of the current input bit being read.

The fact that pseudorandomness for permutation branching programs might be easier than for general branching programs was suggested by the proof that Undirected S-T Connectivity is in Logspace [14] and its follow-ups [15, 30]. Specifically, the latter works construct "pseudorandom walk generators" for "consistently labelled" graphs. Interpreted for permutation branching programs, these results ensure that if an ordered permutation branching program has the property that every layer has a nonnegligible amount of "mixing" — meaning that the distribution on states becomes closer to uniform, on a truly random input — then the overall program will also have mixing when run on the output of the pseudorandom generator (albeit at a slower rate). The generator has a seed length of $O(\log n)$ even for ordered permutation branching programs of width $\mathrm{poly}(n)$. Reingold, Trevisan, and Vadhan [15] also show that if a generator with similar properties could be constructed for (ordered) regular branching programs of polynomial width, then this would suffice to prove RL = L. Thus, in the case of polynomial width, regularity is not a significant constraint.

Recently, there has been substantial progress on constructing pseudorandom generators for ordered regular and permutation branching programs of constant width. Braverman, Rao, Raz, and Yehudayoff [19] and Brody and Verbin [20] gave pseudorandom generators with seed length $\tilde{O}(\log n)$ for ordered regular branching programs of constant width. Koucký, Nimbhorkar and Pudlák [21] showed that the seed length could be further improved to $O(\log n)$ for ordered, permutation branching programs of constant width; see [22, 23] for simplifications and improvements.

All of these generators for ordered regular and permutation branching programs are based on refined analyses of the pseudorandom generator construction of Impagliazzo, Nisan, and Wigderson [24].

1.2 Our Results and Techniques

Our main result is a pseudorandom generator for read-once, oblivious, (unordered) permutation branching programs of constant width:

Theorem 1.1 (Main Result). *For every constant w, there is an explicit pseudorandom generator $G : \{0,1\}^{O(\log^2 n)} \to \{0,1\}^n$ fooling oblivious, read-once (but unordered), permutation branching programs of width w and length n.*

To be precise, the seed length and space complexity of the pseudorandom generator is

$$O(w^2 \log(w) \log(n) \log(nw/\varepsilon) + w^4 \log^2(w/\varepsilon))$$

for oblivious, read-once, permutation branching programs of length n and width w, where ε is the error.

Previously, it was only known how to achieve a seed length of $n^{1/2+o(1)}$ for this model, as follows from the aforementioned results of Impagliazzo, Meka, and Zuckerman [29] (which actually holds for arbitrary branching programs).

Our techniques also achieve seed length $n^{1/2+o(1)}$ for arbitrary read-once, oblivious branching programs of width up to $2^{n^{o(1)}}$:

Theorem 1.2. *There is an explicit pseudorandom generator $G : \{0,1\}^{\tilde{O}(\sqrt{n} \log w)} \to \{0,1\}^n$ fooling oblivious, read-once (but unordered) branching programs of width w and length n.*

This result is incomparable to that of Impagliazzo et al. [29]. Their seed length depends polynomially on the width w, so require width $w = n^{o(1)}$ to achieve seed length $n^{1/2+o(1)}$. On the other hand, our result is restricted to *read-once, oblivious* branching programs.

Our construction of the generator in Theorem 1.1 is essentially the same as the generator of Gopalan et al. [25] for read-once CNF formulas, but with a new analysis (and different setting of parameters) for read-once, oblivious, permutation branching programs. The generator works by selecting a subset $T \subset [n]$ of output coordinates in a pseudorandom way, assigning the bits in T using another pseudorandom distribution X, and then recursively assigning the bits outside T. We generate T using an almost $O(\log n)$-wise independent distribution, including each coordinate $i \in T$ with a constant probability p_w depending only on the width w. We assign the bits in T using a small-bias distribution X on $\{0,1\}^n$ [26]; such a generator has the property that for every nonempty subset $S \subset [n]$, the parity $\oplus_{i \in S} X_i$ of bits in S has bias at most ε. Generating T requires $O(\log n)$ random bits, generating X requires $O(\log n)$ bits (even for $\varepsilon = 1/\text{poly}(n)$), and we need $O(\log n)$ levels of recursion to assign all the bits. This gives us our $O(\log^2 n)$ seed length.

Let $B : \{0,1\}^n \to \{0,1\}$ be a function computed by an oblivious, read-once, permutation branching program of width w. Following [25], to show that our pseudorandom generator fools B, it suffices to show that the partial assignment generated in a single level of recursion approximately preserves the acceptance probability of B (on average). To make this precise, we need a bit of notation.

For a set $t \subset [n]$, a string $x \in \{0,1\}^n$, and $y \in \{0,1\}^{n-|t|}$, define $\mathrm{Select}(t,x,y) \in \{0,1\}^n$ as follows:

$$\mathrm{Select}(t,x,y)_i = \begin{cases} x_i & \text{if } i \in t \\ y_{|\{j \le i : j \notin t\}|} & \text{if } i \notin t \end{cases}$$

Once we choose a set $t \leftarrow T$ and an assignment $x \leftarrow X$ to the variables in t, the residual acceptance probability of B is $\mathop{\mathbb{P}}_{U}[B(\mathrm{Select}(t,x,U)) = 1]$, where U is the uniform distribution on $\{0,1\}^n$. So, the average acceptance probability over $t \leftarrow T$ and $x \leftarrow X$ is $\mathop{\mathbb{P}}_{T,X,U}[B(\mathrm{Select}(T,X,U)) = 1]$. We would like this to be close to the acceptance probability under uniformly random bits, namely $\mathop{\mathbb{P}}_{U}[B(U) = 1] = \mathop{\mathbb{P}}_{T,U',U}[B(\mathrm{Select}(T,U',U) = 1]$. That is, we would like our small-bias distribution X to fool the function $B'(x) := \mathop{\mathbb{E}}_{T,U}[B(\mathrm{Select}(T,x,U))]$. The key insight in [25] is that B' can be a significantly easier function to fool than B, and even than fixed restrictions of B (like $B(\mathrm{Select}(t,\cdot,y))$ for fixed t and y). We show that the same phenomenon holds for oblivious, read-once, *regular* branching programs. (The reason that the analysis of our overall pseudorandom generator applies only for *permutation* branching programs is that regularity is not preserved under restriction (as needed for the recursion), whereas the permutation property is.)

To show that a small-bias space fools $B'(x)$, it suffices to show that the **Fourier mass** of B', namely $\sum_{s \in \{0,1\}^n, s \neq 0} |\widehat{B'}[s]|$, is bounded by $\mathrm{poly}(n)$. (Here $\widehat{B'}[s] = \mathop{\mathbb{E}}_{U}\left[B'[U] \cdot (-1)^{s \cdot U}\right]$ is the standard Fourier transform over $\mathbb{Z}_2^n$. So $\widehat{B'}[s]$ measures the correlation of B' with the parity function defined by s.) We show that this is indeed the case (for most choices of the set $t \leftarrow T$):

Theorem 1.3 (Main Lemma). *For every constant w, there are constants $p_w > 0$ and $d_w \in \mathbb{N}$ such that the following holds. Let $B : \{0,1\}^n \to \{0,1\}$ be computed by an oblivious, read-once, regular branching program of width w and length $n \ge d_w$. Let $T \subset [n]$ be a randomly chosen set so that every coordinate $i \in [n]$ is placed in T with probability p_w and these choices are n^{-d_w}-almost $(d_w \log n)$-wise independent. Then with high probability over $t \leftarrow T$ $B'(x) = \mathop{\mathbb{E}}_{U}[B(\mathrm{Select}(t,x,U))]$ has Fourier mass at most n^{d_w}.*

As a warm-up, we begin by analysing the Fourier mass in the case the set T is chosen completely at random, with every coordinate included independently with probability p_w. In this case, it is more convenient to average over T and work with $B'(x) = \mathop{\mathbb{E}}_{T,U}[B(\mathrm{Select}(T,x,U))]$. Then it turns out that $\widehat{B'}[s] = p_w^{|s|} \cdot \widehat{B}[s]$, where $|s|$ denotes the Hamming weight of the vector s. Thus, it suffices to analyse the original program B and show that for each $k \in \{1, \cdots, n\}$, the Fourier mass of B restricted to s of weight k is at most c_w^k, where c_w is a constant depending only on w (not on n). We prove that this is indeed the case for regular branching programs:

Theorem 1.4. *Let $B : \{0,1\}^n \to \{0,1\}$ be a function computed by an oblivious, read-once, regular branching program of width w. Then for every $k \in \{1,\ldots,n\}$, we have*

$$\sum_{s\in\{0,1\}^n:|s|=k} |\widehat{B}[s]| \leq (2w^2)^k.$$

Our proof of Theorem 1.4 relies on the main lemma of Braverman et al. [19], which intuitively says that in a bounded-width, read-once, oblivious, regular branching program, only a constant number of bits have a significant effect on the acceptance probability. More formally, if we sum, for every time step i and all possible states v at time i, the absolute difference between the acceptance probability after reading a 0 versus reading a 1 from state v, the total will be bounded by $\mathrm{poly}(w)$ (independent of n). This directly implies a bound of $\mathrm{poly}(w)$ on the Fourier mass of B at the first level: the correlation of B with a parity of weight 1 is bounded by the effect of a single bit on the output of B. We then bound the correlation of B with a parity of weight k by the correlation of a *prefix* of B with a parity of weight $k-1$ times the effect of the remaining bit on B. Thus we inductively obtain the bound on the Fourier mass of B at level k.

Our proof of Theorem 1.3 for the case of a pseudorandom restriction T uses the fact that we can decompose the high-order Fourier coefficients of an oblivious, read-once branching program B' into products of low-order Fourier coefficients of "subprograms" (intervals of consecutive layers) of B'. Using an almost $O(\log n)$-wise independent choice of T enables us to control the Fourier mass at level $O(\log n)$ for all subprograms of B', which suffices to control the total Fourier mass of B'.

2 Preliminaries

2.1 Branching Programs

We define a length-n, width-w **program** to be a function $B : \{0,1\}^n \times [w] \to [w]$, which takes a start state $u \in [w]$ and an input string $x \in \{0,1\}^n$ and outputs a final state $B[x](u)$.

In our applications, the input x is randomly (or pseudorandomly) chosen, in which case a program can be viewed as a Markov chain randomly taking initial states to final states. For each $x \in \{0,1\}^n$, we let $B[x] \in \{0,1\}^{w\times w}$ be a matrix defined by

$$B[x](u,v) = 1 \iff B[x](u) = v.$$

For a random variable X on $\{0,1\}^n$, we have $\mathop{\mathbb{E}}_{X}[B[X]] \in [0,1]^{w\times w}$, where $\mathop{\mathbb{E}}_{R}[f(R)]$ is the **expectation** of a function f with respect to a random variable R. Then the entry in the u^{th} row and v^{th} column $\mathop{\mathbb{E}}_{X}[B[X]](u,v)$ is the probability that B takes the initial state u to the final state v when given a random input from the distribution X.

A branching program reads one bit of the input at a time (rather than reading x all at once) maintaining only a state in $[w] = \{1,2,\cdots,w\}$ at each step.

We capture this restriction by demanding that the program be composed of several smaller programs, as follows.

Let B and B' be width-w programs of length n and n' respectively. We define the **concatenation** $B \circ B' : \{0,1\}^{n+n'} \times [w] \to [w]$ of B and B' by

$$(B \circ B')[x \circ x'](u) := B'[x'](B[x](u)),$$

which is a width-w, length-$(n+n')$ program. That is, we run B and B' on separate inputs, but the final state of B becomes the start state of B'. Concatenation corresponds to matrix multiplication—that is, $(B \circ B')[x \circ x'] = B[x] \cdot B'[x']$, where the two programs are concatenated on the left hand side and the two matrices are multiplied on the right hand side.

A length-n, width-w, **ordered branching program** is a program B that can be written $B = B_1 \circ B_2 \circ \cdots \circ B_n$, where each B_i is a length-1 width-w program. We refer to B_i as the i^{th} **layer** of B. We denote the **subprogram** of B from layer i to layer j by $B_{i \cdots j} := B_i \circ B_{i+1} \circ \cdots \circ B_j$.

General read-once, oblivious branching programs (a.k.a. unordered branching programs) can be reduced to the ordered case by a permutation of the input bits. Formally, a **read-once, oblivious branching program** B is an ordered branching program B' composed with a permutation π. That is, $B[x] = B'[\pi(x)]$, where the i^{th} bit of $\pi(x)$ is the $\pi(i)^{\text{th}}$ bit of x.

For a program B and an arbitrary distribution X, the matrix $\mathop{\mathbb{E}}_{X}[B[X]]$ is **stochastic**—that is, $\sum_v \mathop{\mathbb{E}}_{X}[B[X]](u,v) = 1$ for all u and $\mathop{\mathbb{E}}_{X}[B[X]](u,v) \geq 0$ for all u and v. A program B is called a **regular program** if the matrix $\mathop{\mathbb{E}}_{U}[B[U]]$ is **doubly stochastic**—that is, both $\mathop{\mathbb{E}}_{U}[B[U]]$ and its transpose $\mathop{\mathbb{E}}_{U}[B[U]]^*$ are stochastic. A program B is called a **permutation program** if $B[x]$ is a permutation matrix for every x or, equivalently, $B[x]$ is doubly stochastic. Note that a permutation program is necessarily a regular program and, if both B and B' are regular or permutation programs, then so is their concatenation.

A regular program B has the property that the uniform distribution is a stationary distribution of the Markov chain $\mathop{\mathbb{E}}_{U}[B[U]]$, whereas, if B is a permutation program, the uniform distribution is stationary for $\mathop{\mathbb{E}}_{X}[B[X]]$ for *any* X.

A **regular branching program** is a branching program where each layer B_i is a regular program and likewise for a **permutation branching program**.

2.2 Fourier Analysis

Let $B : \{0,1\}^n \to \mathbb{R}^{w \times w}$ be a matrix-valued function (such as given by a length-n, width-w branching program). Then we define the **Fourier transform** of B as a matrix-valued function $\widehat{B} : \{0,1\}^n \to \mathbb{R}^{w \times w}$ given by

$$\widehat{B}[s] := \mathop{\mathbb{E}}_{U}[B[U]\chi_s(U)],$$

where $s \in \{0,1\}^n$ (or, equivalently, $s \subset [n]$) and

$$\chi_s(x) = (-1)^{\sum_i x(i) \cdot s(i)} = \prod_{i \in s} (-1)^{x(i)}.$$

We refer to $\widehat{B}[s]$ as the s^{th} **Fourier coefficient** of B. The **order** of a Fourier coefficient $\widehat{B}[s]$ is $|s|$—the **Hamming weight** of s, which is the size of the set s or the number of 1s in the string s. Note that this is equivalent to taking the real-valued Fourier transform of each of the w^2 entries of B separately, but we will see below that this matrix-valued Fourier transform is nicely compatible with matrix algebra.

For a random variable X over $\{0,1\}^n$ we define its s^{th} **Fourier coefficient** as

$$\widehat{X}(s) := \mathop{\mathbb{E}}_{X} \left[\chi_s(X)\right],$$

which, up to scaling, is the same as taking the real-valued Fourier transform of the probability mass function of X. We have the following useful properties.

Lemma 2.1. *Let $A, B : \{0,1\}^n \to \mathbb{R}^{w \times w}$ be matrix valued functions. Let X, Y, and U be independent random variables over $\{0,1\}^n$, where U is uniform. Let $s, t \in \{0,1\}^n$. Then we have the following.*

- *Decomposition: If $C[x \circ y] = A[x] \cdot B[y]$ for all $x, y \in \{0,1\}^n$, then $\widehat{C}[s \circ t] = \widehat{A}[s] \cdot \widehat{B}[t]$.*
- *Expectation: $\mathop{\mathbb{E}}_{X}[B[X]] = \sum_s \widehat{B}[s]\widehat{X}(s)$.*
- *Parseval's Identity: $\sum_{s \in \{0,1\}^n} \left|\left|\widehat{B}[s]\right|\right|_{Fr}^2 = \mathop{\mathbb{E}}_{U}\left[||B[U]||_{Fr}^2\right]$, where $||\cdot||_{Fr}$ is the Frobenius norm.*

The Decomposition property is what makes the matrix-valued Fourier transform more convenient than separately taking the Fourier transform of the matrix entries as done in [28]. If B is a length-n width-w branching program, then, for all $s \in \{0,1\}^n$,

$$\widehat{B}[s] = \widehat{B}_1[s_1] \cdot \widehat{B}_2[s_2] \cdot \cdots \cdot \widehat{B}_n[s_n].$$

2.3 Fourier Mass

Define the **Fourier mass** of a matrix-valued function B to be

$$L_2(B) := \sum_{s \neq 0} \left|\left|\widehat{B}[s]\right|\right|_2,$$

where $||M||_2 := \max_x ||xM||_2 / ||x||_2$ is the **spectral norm**. Also, define the **Fourier mass of B at level k** as

$$L_2^k(B) := \sum_{s \in \{0,1\}^n : |s| = k} \left|\left|\widehat{B}[s]\right|\right|_2.$$

Note that $L_2(B) = \sum_{k \geq 1} L_2^k(B)$.

The Fourier mass is unaffected by order:

Lemma 2.2. *Let $B, B' : \{0,1\}^n \to \mathbb{R}^{w \times w}$ be matrix-valued functions satisfying $B[x] = B'[\pi(x)]$, where $\pi : [n] \to [n]$ is a permutation. Then, for all $s \in \{0,1\}^n$, $\widehat{B}[s] = \widehat{B'}[\pi(s)]$. In particular, $L_2(B) = L_2(B')$ and $L_2^k(B) = L_2^k(B')$ for all k.*

Lemma 2.2 implies that the Fourier mass of any read-once, oblivious branching program is equal to the Fourier mass of the corresponding ordered branching program.

A random variable X is called **ε-biased** if $|\hat{X}[s]| \leq \varepsilon$ for all $s \neq 0^n$ [26]. If $L_2(B)$ is small, then B is fooled by any small-bias distribution:

Lemma 2.3. *Let B be a length-n, width-w, branching program. Let X be a ε-biased random variable on $\{0,1\}^n$. We have*

$$\left|\left|\mathop{\mathbb{E}}_{X}[B[X]] - \mathop{\mathbb{E}}_{U}[B[U]]\right|\right|_2 = \left|\left|\sum_{s \neq 0} \widehat{B}[s]\widehat{X}(s)\right|\right|_2 \leq L_2(B)\varepsilon.$$

In the worst case $L_2(B) = 2^{\Theta(n)}$, even for a length-n width-3 permutation branching program B. For example, the program $B_{\mathrm{mod}\,3}$ that computes the Hamming weight of its input modulo 3 has exponential Fourier mass.

We show that, using 'restrictions', we can ensure that $L_2(B)$ is small.

3 Fourier Analysis of Regular Branching Programs

We use a result by Braverman et al. [19]. The following is a Fourier-analytic reformulation of their result.

Lemma 3.1 ([19, Lemma 4]). *Let B be a length-n, width-w, ordered, regular branching program. Then*

$$\sum_{1 \leq i \leq n} \left|\left|\widehat{B_{i \cdots n}}[1 \circ 0^{n-i}]\right|\right|_2 \leq 2w^2.$$

Braverman et al. instead consider the sum, over all $i \in [n]$ and all states $u \in [w]$ at layer i, of the difference in acceptance probabilities if we run the program starting at v with a 0 followed by random bits versus a 1 followed by random bits. They refer to this quantity as the **weight** of B. Their result can be expressed in Fourier-analytic terms by considering subprograms $B_{i \cdots n}$ that are the original program with the first $i-1$ layers removed:

$$\sum_{1 \leq i \leq n} \left|\left|\widehat{B_{i \cdots n}}[1 \circ 0^{n-i}]q\right|\right|_1 \leq 2(w-1)$$

for any $q \in \{0,1\}^w$ with $\sum_u q(u) = 1$. (The vector q can be used to specify the accept state of B, and the v^{th} row of $\widehat{B_{i \cdots n}}[1 \circ 0^{n-i}]q$ is precisely the difference

in acceptance probabilities mentioned above.) By summing over all w possible q, we obtain

$$\sum_{i\in[n]}\sum_{u}\left|\left|\widehat{B_{i\cdots n}}[1\circ 0^{n-i}](\cdot,u)\right|\right|_1 \le 2w(w-1).$$

This implies Lemma 3.1, as the spectral norm of a matrix is bounded by the sum of the 1-norms of the columns.

Lemma 3.1 is similar (but not identical) to a bound on the first-order Fourier coefficients of a regular branching program: The term $\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]$ measures the effect of the i^{th} bit on the output of B when we start the program at layer i, whereas the i^{th} first-order Fourier coefficient $\widehat{B}[0^{i-1}\circ 1\circ 0^{n-i}]$ measures the effect of the i^{th} bit when we start at the first layer and run the first $i-1$ layers with random bits. This difference allows us to use Lemma 3.1 to obtain a bound on all low-order Fourier coefficients of a regular branching program:

Theorem 3.2. *Let B be a length-n, width-w, read-once, oblivious, regular branching program. Then*

$$L_2^k(B) := \sum_{s\in\{0,1\}^n:|s|=k}\left|\left|\widehat{B}[s]\right|\right|_2 \le (2w^2)^k.$$

The bound does not depend on n, even though we are summing $\binom{n}{k}$ terms.

Proof. By Lemma 2.2, we may assume that B is ordered. We perform an induction on k. If $k=0$, then there is only one Fourier coefficient to bound—namely, $\widehat{B}[0^n] = \mathop{\mathbb{E}}_{U}[B[U]]$, which is doubly stochastic. The base case follows from the fact that every doubly stochastic matrix has spectral norm 1. Suppose the result holds for k. We split the Fourier coefficients based on where the last 1 is:

$$\begin{aligned}
&\sum_{s\in\{0,1\}^n:|s|=k+1}\left|\left|\widehat{B}[s]\right|\right|_2\\
&\quad=\sum_{1\le i\le n}\sum_{s\in\{0,1\}^{i-1}:|s|=k}\left|\left|\widehat{B}[s\circ 1\circ 0^{n-i}]\right|\right|_2\\
&\quad\le\sum_{1\le i\le n}\sum_{s\in\{0,1\}^{i-1}:|s|=k}\left|\left|\widehat{B_{1\cdots i-1}}[s]\right|\right|_2\cdot\left|\left|\widehat{B_{i\cdots n}}[1\circ 0^{n-i}]\right|\right|_2\\
&\qquad\text{(by Lemma 2.1 (Decomposition))}\\
&\quad\le(2w^2)^k\cdot 2w^2\quad\text{(by the induction hypothesis and Lemma 3.1).}
\end{aligned}$$

4 Random Restrictions

Our results involve restricting branching programs. However, our use of restrictions is different from elsewhere in the literature. Here, as in [25], we use (pseudorandom) restrictions in the usual way, but we *analyse* them by averaging over the *unrestricted* bits. Formally, we define a restriction as follows.

Definition 4.1. *For $t \in \{0,1\}^n$ and a length-n branching program B, let $B|_t$ be the **restriction** of B to t—that is, $B|_t : \{0,1\}^n \to \mathbb{R}^{w \times w}$ is a matrix-valued function given by $B|_t[x] := \mathop{\mathbb{E}}_U [B[\mathrm{Select}(t, x, U)]]$, where U is uniform on $\{0,1\}^n$.*

The most important aspect of restrictions is how they relate to the Fourier transform: For all B, s, and t, we have $\widehat{B|_t}[s] = \widehat{B}[s]$ if $s \subset t$ and $\widehat{B|_t}[s] = 0$ otherwise. The restriction t 'kills' all the Fourier coefficients that are not contained in it. This means that a restriction significantly reduces the Fourier mass:

Lemma 4.2. *Let B be a length-n, width-w program. Let T be n independent random bits each with expectation p. Then*

$$\mathop{\mathbb{E}}_T [L_2(B|_T)] = \sum_{s \neq 0} p^{|s|} \left|\left| \widehat{B}[s] \right|\right|_2 .$$

We can use Theorem 3.2 to prove a result about random restrictions of regular branching programs:

Proposition 4.3. *Let B be a length-n, width-w, read-once, oblivious, regular branching program. Let T be n independent random bits each with expectation $p \leq 1/4w^2$. Then $\mathop{\mathbb{E}}_T [L_2(B|_T)] \leq 1$.*

5 Pseudorandom Restrictions

To analyse our generator, we need a pseudorandom version of Proposition 4.3. That is, we need to prove that, for a *pseudorandom* T (generated using few random bits), $L_2(B|_T)$ is small. We will generate T using an almost $O(\log n)$-wise independent distribution:

Definition 5.1. *A random variable X on Ω^n is δ-**almost** k-**wise independent** if, for any $I = \{i_1, i_2, \cdots, i_k\} \subset [n]$ with $|I| = k$, the coordinates $(X_{i_1}, \cdots, X_{i_k}) \in \Omega^k$ are δ statistically close to being independent. We say that X is k-**wise independent** if it is 0-almost k-wise independent.*

We can sample a random variable X on $\{0,1\}^n$ that is δ-almost k-wise independent such that each bit has expectation $p = 2^{-d}$ using $O(kd + \log(1/\delta) + d\log(nd))$ random bits. See the full version of this paper for more details.

Our main lemma (stated informally as Theorem 1.3) is as follows.

Theorem 5.2 (Main Lemma). *Let B be a length-n, width-w, read-once, oblivious, regular branching program. Let T be a random variable over $\{0,1\}^n$ where each bit has expectation p and the bits are δ-almost $2k$-wise independent. Suppose $p \leq (2w)^{-2}$ and $\delta \leq (2w)^{-4k}$. Then*

$$\mathop{\mathbb{P}}_T \left[L_2(B|_T) \leq (2w^2)^k \right] \geq 1 - n^4 \cdot \frac{2}{2^k}.$$

In particular, we show that, for $w = O(1)$, $k = O(\log n)$, and $\delta = 1/\mathrm{poly}(n)$, we have $L_2(B|_T) \leq \mathrm{poly}(n)$ with probability $1 - 1/\mathrm{poly}(n)$.

First we show that the Fourier mass at level $O(\log n)$ is bounded by $1/n$ with high probability. This also applies to all subprograms.

Lemma 5.3. *Let B be a length-n, width-w, ordered, regular branching program. Let T be a random variable over $\{0,1\}^n$ where each bit has expectation p and the bits are δ-almost k-wise independent. If $p \leq (2w)^{-2}$ and $\delta \leq (2w)^{-2k}$, then, for all $\beta > 0$,*

$$\mathop{\mathbb{P}}_{T}\left[\forall 1 \leq i \leq j \leq n \quad L_2^k(B_{i\cdots j}|_T) \leq \beta\right] \geq 1 - n^2 \frac{2}{2^k \beta}.$$

Proof. By Theorem 3.2, for all i and j,

$$\mathop{\mathbb{E}}_{T}\left[L_2^k(B_{i\cdots j}|_T)\right] = \sum_{s \subset \{i\cdots j\} : |s| = k} \mathop{\mathbb{P}}_{T}[s \subset T] \left|\left|\widehat{B_{i\cdots j}}[s]\right|\right|_2 \leq (2w^2)^k (p^k + \delta) \leq \frac{2}{2^k}.$$

The result now follows from Markov's inequality and a union bound.

Now we use Lemma 5.3 to bound the Fourier mass at higher levels. We decompose high-order ($k' \geq 2k$) Fourier coefficients into low-order ($k \leq k' < 2k$) ones, similarly to the proof of Theorem 3.2:

Lemma 5.4. *Let B be a length-n, ordered branching program and $t \in \{0,1\}^n$. Suppose that, for all i, j, and k' with $1 \leq i \leq j \leq n$ and $k \leq k' < 2k$, $L_2^{k'}(B_{i\cdots j}|_t) \leq 1/n$. Then, for all $k'' \geq k$ and all i and j, $L_2^{k''}(B_{i\cdots j}|_t) \leq 1/n$.*

Lemmas 5.3 and 5.4 combine to give Theorem 5.2: By Lemma 5.3, a pseudorandom restriction guarantees that, with high probability the Fourier mass at levels k to $2k$ is small for all subprograms $B_{i\cdots j}$. Lemma 5.4 implies that, with high probability, the Fourier mass is small at all levels above k. The Fourier mass at levels below k can be bounded directly using Theorem 3.2.

6 The Pseudorandom Generator

Our main result (Theorem 1.1) is stated more formally as follows.

Theorem 6.1 (Main Result). *There exists a pseudorandom generator family $G_{n,w,\varepsilon} : \{0,1\}^{s_{n,w,\varepsilon}} \to \{0,1\}^n$ with seed length*

$$s_{n,w,\varepsilon} = O(w^2 \log(w) \log(n) \log(nw/\varepsilon) + w^4 \log^2(w/\varepsilon))$$

such that, for any length-n, width-w, read-once, oblivious (but unordered), permutation branching program B and $\varepsilon > 0$,

$$\left|\left| \mathop{\mathbb{E}}_{U_{s_{n,w,\varepsilon}}} \left[B[G_{n,w,\varepsilon}(U_{s_{n,w,\varepsilon}})]\right] - \mathop{\mathbb{E}}_{U}[B[U]] \right|\right|_2 \leq \varepsilon.$$

Moreover, $G_{n,w,\varepsilon}$ can be computed in space $O(s_{n,w,\varepsilon})$.

Now we use the above results to construct our pseudorandom generator for a read-once, oblivious, permutation branching program B.

Theorem 5.2 says that with high probability over T, $B|_T$ has small Fourier mass, where T is almost k-wise independent with each bit having expectation p. This implies that $B|_T$ is fooled by small bias X and thus

$$\mathop{\mathbb{E}}_{T,X,U}[B[\mathrm{Select}(T,X,U)]] \approx \mathop{\mathbb{E}}_{T,U,U'}[B[\mathrm{Select}(T,U',U)]] = \mathop{\mathbb{E}}_{U}[B[U]]\,.$$

If we define $\overline{B}_{t,x}[y] := B[\mathrm{Select}(t,x,y)]$, then $\mathop{\mathbb{E}}_{T,X,U}\left[\overline{B}_{T,X}[U]\right] \approx \mathop{\mathbb{E}}_{U}[B[U]]$. So now we need only construct a pseudorandom generator for $\overline{B}_{t,x}$, which is a length-$(n-|t|)$ permutation branching program. Then

$$\mathop{\mathbb{E}}_{T,X,\tilde{U}}\left[\overline{B}_{T,X}[\tilde{U}]\right] \approx \mathop{\mathbb{E}}_{T,X,U}\left[\overline{B}_{T,X}[U]\right] \approx \mathop{\mathbb{E}}_{U}[B[U]]\,,$$

where $\tilde{U}$ is the output of the pseudorandom generator for $\overline{B}_{t,x}$. We construct $\tilde{U} \in \{0,1\}^{n-|T|}$ recursively; each time we recurse, the required output length is reduced to $n - |T| \approx n(1-p)$. Thus after $O(\log(n)/p)$ levels of recursion the required output length is constant.

The only place where the analysis breaks down for regular branching programs is when we recurse. If B is only a *regular* branching program, $\overline{B}_{t,x}$ may not be regular. However, if B is a *permutation* branching program, then $\overline{B}_{t,x}$ is too. Essentially, the only obstacle to generalising the analysis to regular branching programs is that regular branching programs are not closed under restrictions.

The pseudorandom generator is formally defined as follows.

Algorithm for $G_{n,w,\varepsilon} : \{0,1\}^{s_{n,w,\varepsilon}} \to \{0,1\}^n$.

1. Compute appropriate values of $p \in [1/8w^2, 1/4w^2]$, $k \geq \log_2\left(4\sqrt{w}n^4/\varepsilon\right)$, $\delta = \varepsilon(2w)^{-4k}$, and $\mu = \varepsilon(2w^2)^{-k}$.[1]
2. If $n \leq (4 \cdot \log_2(2/\varepsilon)/p)^2$, output n truly random bits and stop.
3. Sample $T \in \{0,1\}^n$ where each bit has expectation p and the bits are δ-almost $2k$-wise independent.
4. If $|T| < pn/2$, output 0^n and stop.
5. Recursively sample $\tilde{U} \in \{0,1\}^{\lfloor n(1-p/2)\rfloor}$. i.e. $\tilde{U} = G_{\lfloor n(1-p/2)\rfloor,w,\varepsilon}(U)$.
6. Sample $X \in \{0,1\}^n$ from a μ-biased distribution.
7. Output $\mathrm{Select}(T,X,\tilde{U}) \in \{0,1\}^n$.

The analysis of the algorithm proceeds as follows.

- Every time we recurse, n is decreased to $\lfloor n(1-p/2)\rfloor$. After $O(\log(n)/p)$ recursions, n is reduced to $O(1)$ and the recursion terminates.
- The probability of failing because $|T| < pn/2$ is small by a Chernoff bound for limited independence. This requires that n is not too small (step 2).
- The output is pseudorandom, as

[1] For the purposes of the analysis we assume that p, k, δ, and μ are the same at every level of recursion. So if $G_{n,w,\varepsilon}$ is being called recursively, use the same values of p, k, δ, and μ as at the previous level of recursion.

$$\mathop{\mathbb{E}}_{T,X,\tilde{U}}\left[B[\mathrm{Select}(T,X,\tilde{U})]\right] \approx \mathop{\mathbb{E}}_{T,X,U}\left[B[\mathrm{Select}(T,X,U)]\right] \approx \mathop{\mathbb{E}}_{U}\left[B[U]\right].$$

The first approximate equality holds because we inductively assume that $\tilde{U}$ is pseudorandom; the second holds as a result of the main lemma.

– The total seed length is the seed length needed to sample X and T at each level of recursion and $O((\log(1/\varepsilon)/p)^2)$ truly random bits at the last level. Sampling X requires seed length $O(\log(n/\mu))$ and sampling T requires seed length $O(k\log(1/p)+\log(\log(n)/\delta))$.

For more details, see the full version of this paper.

7 General Read-Once, Oblivious Branching Programs

With a different setting of parameters, our pseudorandom generator can fool arbitrary oblivious, read-once branching programs, rather than just permutation branching programs (Theorem 1.2). The key to proving Theorem 1.2 is the following Fourier mass bound for arbitrary branching programs.

Lemma 7.1. *Let B be a length-n, width-w, read-once, oblivious branching program. Then, for all $k \in [n]$, $L_2^k(B) \le \sqrt{wn^k}$.*

Proof. By Parseval's Identity,

$$\sum_{s\in\{0,1\}^n:|s|=k}\left|\left|\widehat{B}[s]\right|\right|_2^2 \le \sum_{s\in\{0,1\}^n}\left|\left|\widehat{B}[s]\right|\right|_{\mathrm{Fr}}^2 = \mathop{\mathbb{E}}_{U}\left[||B[U]||_{\mathrm{Fr}}^2\right] = w.$$

The result follows from Cauchy-Schwartz.

For more details, see the full version of this paper.

References

1. Indyk, P.: Stable distributions, pseudorandom generators, embeddings, and data stream computation. J. ACM 53(3), 307–323 (2006)
2. Sivakumar, D.: Algorithmic derandomization via complexity theory. In: CCC 2002, p. 10 (2002)
3. Celis, L.E., Reingold, O., Segev, G., Wieder, U.: Balls and bins: Smaller hash families and faster evaluation. In: FOCS 2011, pp. 599–608 (2011)
4. Healy, A., Vadhan, S., Viola, E.: Using nondeterminism to amplify hardness. SIAM Journal on Computing 35(4), 903–931 (2006)
5. Kaplan, E., Naor, M., Reingold, O.: Derandomized constructions of k-wise (almost) independent permutations. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX 2005 and RANDOM 2005. LNCS, vol. 3624, pp. 354–365. Springer, Heidelberg (2005)
6. Haitner, I., Harnik, D., Reingold, O.: On the power of the randomized iterate. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 22–40. Springer, Heidelberg (2006)

7. Nisan, N.: $\mathcal{RL} \subset \mathcal{SC}$. In: STOC 1992, pp. 619–623. ACM (1992)
8. Nisan, N., Zuckerman, D.: More deterministic simulation in logspace. In: STOC 1993, pp. 235–244. ACM (1993)
9. Raz, R., Reingold, O.: On recycling the randomness of states in space bounded computation. In: STOC 1999, pp. 159–168 (1999)
10. Even, G., Goldreich, O., Luby, M., Nisan, N., Velickovic, B.: Efficient approximation of product distributions. Random Struct. Algorithms 13(1), 1–16 (1998)
11. Linial, N., Luby, M., Saks, M.E., Zuckerman, D.: Efficient construction of a small hitting set for combinatorial rectangles in high dimension. Combinatorica 17(2), 215–234 (1997)
12. Armoni, R., Saks, M.E., Wigderson, A., Zhou, S.: Discrepancy sets and pseudorandom generators for combinatorial rectangles. In: FOCS 1996, pp. 412–421 (1996)
13. Lu, C.J.: Improved pseudorandom generators for combinatorial rectangles. Combinatorica 22(3), 417–434 (2002)
14. Reingold, O.: Undirected connectivity in log-space. J. ACM 55(4), 17:1–17:24 (2008)
15. Reingold, O., Trevisan, L., Vadhan, S.: Pseudorandom walks on regular digraphs and the $\mathcal{RL}$ vs. $\mathcal{L}$ problem. In: STOC 2006, pp. 457–466. ACM (2006)
16. Saks, M., Zhou, S.: $\mathrm{BP_HSPACE}(S) \subset \mathrm{DSPACE}(S^{3/2})$. J. CSS 58(2), 376–403 (1999)
17. Bogdanov, A., Dvir, Z., Verbin, E., Yehudayoff, A.: Pseudorandomness for width 2 branching programs. ECCC TR09-070 (2009)
18. Šíma, J., Žák, S.: A sufficient condition for sets hitting the class of read-once branching programs of width 3. In: Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turán, G. (eds.) SOFSEM 2012. LNCS, vol. 7147, pp. 406–418. Springer, Heidelberg (2012)
19. Braverman, M., Rao, A., Raz, R., Yehudayoff, A.: Pseudorandom generators for regular branching programs. In: FOCS 2010, pp. 40–47 (2010)
20. Brody, J., Verbin, E.: The coin problem and pseudorandomness for branching programs. In: FOCS 2010, pp. 30–39 (2010)
21. Koucký, M., Nimbhorkar, P., Pudlák, P.: Pseudorandom generators for group products. In: STOC 2011, pp. 263–272. ACM (2011)
22. De, A.: Pseudorandomness for permutation and regular branching programs. In: CCC 2011, pp. 221–231 (2011)
23. Steinke, T.: Pseudorandomness for permutation branching programs without the group theory. ECCC TR12-083 (2012)
24. Impagliazzo, R., Nisan, N., Wigderson, A.: Pseudorandomness for network algorithms. In: STOC 1994, pp. 356–364 (1994)
25. Gopalan, P., Meka, R., Reingold, O., Trevisan, L., Vadhan, S.: Better pseudorandom generators from milder pseudorandom restrictions. In: FOCS 2012, pp. 120–129 (2012)
26. Naor, J., Naor, M.: Small-bias probability spaces: Efficient constructions and applications. SIAM J. Comput. 22, 838–856 (1993)
27. Tzur, Y.: Notions of weak pseudorandomness and $\mathrm{GF}(2^n)$-polynomials. Master's thesis, Weizmann Institute of Science (2009)
28. Bogdanov, A., Papakonstantinou, P.A., Wan, A.: Pseudorandomness for read-once formulas. In: FOCS 2011, pp. 240–246 (2011)
29. Impagliazzo, R., Meka, R., Zuckerman, D.: Pseudorandomness from shrinkage. In: FOCS 2012, pp. 111–119 (2012)
30. Rozenman, E., Vadhan, S.: Derandomized squaring of graphs. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX and RANDOM 2005. LNCS, vol. 3624, pp. 436–447. Springer, Heidelberg (2005)

Absolutely Sound Testing of Lifted Codes

Elad Haramaty[1,⋆], Noga Ron-Zewi[1,⋆⋆], and Madhu Sudan[2]

[1] Department of Computer Science, Technion, Haifa
{eladh,nogaz}@cs.technion.ac.il
[2] Microsoft Research New England, Cambridge, MA
madhu@mit.edu

Abstract. In this work we present a strong analysis of the testability of a broad, and to date the most interesting known, class of "affine-invariant" codes. Affine-invariant codes are codes whose coordinates are associated with a vector space and are invariant under affine transformations of the coordinate space. Affine-invariant linear codes form a natural abstraction of algebraic properties such as linearity and low-degree, which have been of significant interest in theoretical computer science in the past. The study of affine-invariance is motivated in part by its relationship to property testing: Affine-invariant linear codes tend to be locally testable under fairly minimal and almost necessary conditions.

Recent works by Ben-Sasson et al. (CCC 2011) and Guo et al. (ITCS 2013) have introduced a new class of affine-invariant linear codes based on an operation called "lifting". Given a base code over a t-dimensional space, its m-dimensional lift consists of all words whose restriction to every t-dimensional affine subspace is a codeword of the base code. Lifting not only captures the most familiar codes, which can be expressed as lifts of low-degree polynomials, it also yields new codes when lifting "medium-degree" polynomials whose rate is better than that of corresponding polynomial codes, and all other combinatorial qualities are no worse.

In this work we show that codes derived from lifting are also testable in an "absolutely sound" way. Specifically, we consider the natural test: Pick a random affine subspace of base dimension and verify that a given word is a codeword of the base code when restricted to the chosen subspace. We show that this test accepts codewords with probability one, while rejecting words at constant distance from the code with constant probability (depending only on the alphabet size). This work thus extends the results of Bhattacharyya et al. (FOCS 2010) and Haramaty et al. (FOCS 2011), while giving concrete new codes of higher rate that have absolutely sound testers. In particular we show that there exists codes satisfying the requirements of Barak et al. (FOCS 2012) to construct small set expanders with a large number of eigenvalues close to the maximal one, with rate slightly higher than the ones used in their work.

⋆ Research was conducted while the author was an intern at Microsoft Research New-England, Cambridge, MA.

⋆⋆ Research was conducted in part while the author was visiting Microsoft Research New-England, Cambridge, MA, and supported in part by a scholarship from the Israel Ministry of Science and Technology. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 671–682, 2013.

1 Introduction

In this work we present results on the testability of "affine-invariant linear codes". We start with some basic terminology before describing our work in greater detail.

Let $\mathbb{F}_q$ denote the finite field of q elements and $\{\mathbb{F}_q^n \to \mathbb{F}_q\}$ denote the set of functions mapping $\mathbb{F}_q^n$ to $\mathbb{F}_q$. In this work a code (or a family) will be a subset of functions $\mathcal{F} \subseteq \{\mathbb{F}_q^n \to \mathbb{F}_q\}$. We use $\delta(f,g)$ to denote the normalized Hamming distance between f and g, i.e., the fraction of inputs $x \in \mathbb{F}_q^n$ for which $f(x) \neq g(x)$. We use $\delta(\mathcal{F})$ to denote $\min_{f \neq g, f,g \in \mathcal{F}}\{\delta(f,g)\}$ and $\delta_{\mathcal{F}}(f)$ to denote $\min_{g \in \mathcal{F}}\{\delta(f,g)\}$. A code $\mathcal{F}$ is said to be a linear code if it is an $\mathbb{F}_q$-subspace, i.e., for every $\alpha \in \mathbb{F}_q$ and $f, g \in \mathcal{F}$, we have $\alpha f + g \in \mathcal{F}$. A function $T : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is said to be an affine transformation if there exists a matrix $B \in \mathbb{F}_q^{n \times n}$ and vector $c \in \mathbb{F}_q^n$ such that $T(x) = Bx + c$. The code $\mathcal{F} \subseteq \{\mathbb{F}_q^n \to \mathbb{F}_q\}$ is said to be affine-invariant if for every affine transformation T and every $f \in \mathcal{F}$ we have $f \circ T \in \mathcal{F}$ (where $(f \circ T)(x) = f(T(x))$).

Affine-invariant linear codes form a very natural abstraction of the class of low-degree polynomials: The set of polynomials of degree at most d is a linear subspace and is closed under affine transformations. Furthermore, as shown by Kaufman and Sudan [16] affine-invariant linear codes retain some of the "locality" properties of multivariate polynomial codes (or Reed-Muller codes), such as local testability and local decodability, that have found many applications in computational complexity. This has led to a sequence of works exploring these codes, but most of the works led to codes of smaller rate than known ones, or gave alternate understanding of known codes [9,10,6,5,4]. A recent work by Guo et al. [11] however changes the picture significantly. They study a "lifting" operator on codes and show that it leads to codes with, in some cases dramatic, improvement in parameters compared to Reed-Muller codes. Our work complements theirs by showing that one family of "best-known" tests manages to work abstractly for codes developed by lifting.

We start by describing the lifting operation: Roughly a lifting of a base code leads to a code in more variables whose codewords are words of the base code on every affine subspace of the base dimension. We define this formally next. For $f : \mathbb{F}_q^n \to \mathbb{F}_q$ and $S \subseteq \mathbb{F}_q^n$, let $f|_S$ denote the restriction of f to the set S. A set $A \subseteq \mathbb{F}_q^n$ is said to be a t-dimensional affine subspace, if there exist $\alpha_0, \ldots, \alpha_t \in \mathbb{F}_q^n$ such that $A = \{\alpha_0 + \sum_{i=1}^t \alpha_i x_i | x_1, \ldots, x_t \in \mathbb{F}_q\}$. We use some arbitrary $\mathbb{F}_q$-linear isomorphism from A to $\mathbb{F}_q^t$ to view $f|_A$ as a function from $\{\mathbb{F}_q^t \to \mathbb{F}_q\}$. Given an affine-invariant linear base code $\mathcal{B} \subseteq \{\mathbb{F}_q^t \to \mathbb{F}_q\}$ and integer $n \geq t$, the *n-dimensional lift* of $\mathcal{B}$, denoted $\mathrm{Lift}_n(\mathcal{B})$, is the set $\{f : \mathbb{F}_q^n \to \mathbb{F}_q \mid f|_A \in \mathcal{B}$ for every t-dimensional affine subspace $A \subseteq \mathbb{F}_q^n\}$.

The lifting operation was introduced by Ben-Sasson et al. [5] as a way to build new affine-invariant linear codes that were *not* locally testable. Their codes were also of much lower rate than known affine-invariant linear codes of similar distance. However in more recent work, Guo et al. [11], showed that lifting could be used positively: They used it to build codes with very good locality properties (especially decodability) with rate much better than known affine-invariant linear

ones, and matching qualitatively the performance of the best known codes. Our work attempts to complement their work by showing that these codes, over constant sized alphabets, can be "locally tested" as efficiently as polynomial codes.

Testing and Absolutely Sound Testing. A code $\mathcal{F} \subseteq \{\mathbb{F}_q^n \to \mathbb{F}_q\}$ is said to be a (k, ϵ, δ)-locally testable code (LTC), if $\delta(\mathcal{F}) \geq \delta$ and there exists a probabilistic oracle algorithm that, on oracle access to $f : \mathbb{F}_q^n \to \mathbb{F}_q$, makes at most k queries to f and accepts $f \in \mathcal{F}$ with probability one, while rejecting $f \notin \mathcal{F}$ with probability at least $\epsilon\delta_{\mathcal{F}}(f)$.

For an ensemble of codes $\{\mathcal{F}_m \subseteq \{\mathbb{F}_q^{n_m} \to \mathbb{F}_q\}\}_m$ for infinitely many m, with $\mathcal{F}_m$ being a $(k(m), \epsilon(m), \delta(m))$-LTC, we say that the code has an *absolutely sound* tester if there exists $\epsilon > 0$ such that $\epsilon(m) \geq \epsilon$ for every m.

Any tester can be converted into an absolutely sound one by repeating the test $1/\epsilon(m)$ times. However this comes with an increase in the query complexity (the parameter $k(m)$) and so it makes sense to ask what is the minimum k one can get for an absolutely sound test.

Previous works by Bhattacharyya et al. [7] and Haramaty et al. [13] raised this question in the context of multivariate polynomial codes (Reed-Muller codes) and showed that the "natural tester" for multivariate polynomial codes is absolutely sound, without any repetitions! The natural test here is derived as follows for prime fields:

> To test if a function f is a polynomial of degree at most d, let t be the smallest integer such that there exist functions of degree greater than d in t variables. Pick a random t-dimensional affine subspace A and verify that $f|_A$ is a degree d polynomial.

The natural test thus makes roughly $q^t = q^{(d+1)/(q-1)}$ queries. This number turns out to be optimal for prime fields in that every function looks like a degree d polynomial if queried at at most q^{t-1} points. Such optimal analyses of low-degree tests turn out to have some uses in computational complexity: In particular one of the many ingredients in the elegant constructions of Barak et al. [3] is the absolutely sound analysis of the polynomial codes over $\mathbb{F}_2$.

Returning to the natural test above, it ends being a little less natural, and not quite optimal when dealing with non-prime fields. Turns out one needs to use a larger value of t than the one in the definition above (specifically, $t = q^{(d+1)/(q-q/p)}$ where p is the characteristic of the field $\mathbb{F}_q$). While it is unclear if sampling all the points in the larger dimensional space is really necessary for absolutely sound testing the results so far seem to suggest working with prime fields is a better option.

1.1 Our Work: Motivation and Results

The motivation for our work is two-fold: Our first motivation is to understand "low-degree testing" better. Low-degree testing has played a fundamental role in computational complexity and yet its proofs are barely understood. They tend to

involve a mix of probabilistic, algebraic, and geometric arguments, and the only setting where the mix of these features seems applicable seems to be the setting of low-degree polynomials. Affine-invariant codes naturally seperate the geometry of subspaces in high-dimensional spaces, from the algebra of polynomials of low-degree. Thus extending a proof or analysis method from the setting of low-degree polynomials to the setting of generic geometric arguments has the nice feature that it has the potential to separate the geometric arguments from the algebraic ones.

Within the theme of low-degree testing, the previous works have revealed interesting analyses. And several of these variations in the resulting theorems have played a role in construction of efficient PCPs or more recently in other searches for explicit objects. In particular the literature includes tests such as those originally given by Blum, Luby and Rubinfeld [8] for testing linearity and followed by [20,1,15,14] for testing higher degree polynomials. The aspects of this family of tests are well abstracted in Kaufman and Sudan [16]. But the literature contains other very interesting theorems, such as those of Raz and Safra [18] and Arora and Sudan [2] which tend to work in the "list-decoding" regime. The analysis of the former in particular seems especially amenable to a "generic proof" in the affine-invariant setting and yet such a proof is not yet available. Our work explores a third such paradigm in the analysis of low-degree tests, which was introduced in the above-mentioned "absolutely-sound testers" of Bhattacharyya et al. and Haramaty et al.

Our work starts by noticing that the natural tests above are really "lifting tests": Namely, the test could be applied to any code that is defined as the lift of a base code with the test checking if a given function is a codeword of the base code when restricted to a random small dimensional affine subspace of the base dimension. Indeed this is the natural way of interpreting almost all the previous results in low-degree testing (with the exception of that of [19]). If so, it is natural to ask if the analysis can be carried out to show the absolute soundness of such tests.

The second, more concrete, motivation for our work is the work of Guo et al. [11]. Over prime fields, it was well-known that lifts of low-degree polynomials lead only to polynomials of the same degree (in more variables). Guo et al. show that lifting over non-prime fields leads to better codes than over prime fields! (Prior to their work, it seemed that working with non-prime fields was worse than working with prime fields.) The improved rate gives motivation to study lifted codes in general, and in particular one class of results that would have been nice to extend was the absolutely-sound tester of [13].

In this work we show that the natural test of lifted codes is indeed absolutely sound. The following theorem spells this statement out precisely.

Theorem 1 (Main). *For every prime power q, there exists $\epsilon_q > 0$ such that the following holds: Let $t \leq n$ be positive integers and let $\mathcal{B} \subsetneq \{\mathbb{F}_q^t \to \mathbb{F}_q\}$ be any affine-invariant linear code. Then $\mathcal{F} = \mathrm{Lift}_n(\mathcal{B})$ is $(q^t, \epsilon_q, q^{-t})$-locally testable.*

We stress that the importance of the above is in the absolute soundness, i.e., the fact that ϵ_q does not depend on t or $\mathcal{B}$. If one is willing to let ϵ_q depend on t and $\mathcal{B}$ then such a result follows from the main theorem of [16].

Our result also sets into proper light the previous work of Haramaty et al. [13] who show that the "natural test" for degree d polynomials over the field $\mathbb{F}_q$ of characteristic p makes $q^{(d+1)/(q-q/p)}$ queries and is absolutely sound. Our result does not mention any dependence on p, the characteristic of the field. It turns out that such a dependence comes due to the following proposition.

Let $\mathrm{RM}(n, d, q)$ denote the set of polynomials over $\mathbb{F}_q$ of degree at most d in n variables.

Proposition 1. *For positive integers d and q where q is a power of a prime p, let $t = t_{d,q} = \lceil \frac{d+1}{q-q/p} \rceil$. Then for every $n \geq t$, the Reed-Muller code* $\mathrm{RM}(n, d, q)$ *equals the code* $\mathrm{Lift}_n(\mathrm{RM}(t, d, q))$.

Applying Theorem 1 to $\mathrm{RM}(n, d, q)$ we immediately obtain the main results of [7] and [13]. And the somewhat cumbersome dependence on the characteristic of q can be blamed on the proposition above, rather than any weakness of the testing analysis. Furthermore, as is exploited by Guo et al. [11] if one interprets the proposition above correctly, then one should use lifts of Reed-Muller codes over non-prime fields with dimension being smaller than $t_{d,q}$. These will yield codes of higher rate while our main theorem guarantees that testability does not suffer.

One concrete consequence of our result is in the use of Reed-Muller codes in the work of Barak et al. [3]. They show how to construct small-set expander graphs with many large eigenvalues and one of the ingredients in their result is a tester of Reed-Muller codes over $\mathbb{F}_2$ (codes obtained by lifting an appropriate family of base codes over $\mathbb{F}_2$). Till this work, the binary Reed-Muller code seemed to be the only code with performance good enough to derive their result. Our work shows that using codes over $\mathbb{F}_4$ or $\mathbb{F}_8$ (or any constant power of two) would serve their purpose at least as well, and even give slight (though really negligible) improvements. We elaborate on these codes and their exact parameters in Section 3. (In particular, see Theorem 3.)

Finally, unlike the works of Bhattacharyya et al., and Haramaty et al., we can not claim that our testers are "optimal". This is not because of a weakness in our analysis, rather it is due to the generality of our theorem. For some codes, including the codes considered in the previous works, our theorem is obviously optimal (being the same test and more or less same analysis as previously). Other codes however may possess special properties making them testable much better. In such cases we can not rule out better tests, though we hope our techniques will still be of some use in analyzing tests for such codes.

Future research directions. As noted earlier, the field of low-degree testing has seen several different themes in the analyses. Combined with the work of Kaufman and Sudan [17] our work points to the possibility that much of that study can be explained in terms of the geometry of affine-invariance, and the role of algebra can be encapsulated away nicely. One family of low-degree tests that would

be very nice to include in this general view would be that of Raz and Safra [18]. Their work presents a very general proof technique that uses really little algebra; and seems ideally amenable to extend to the affine-invariant setting. We hope that future work will address this.

We also hope that future work improve the dependence of ϵ_q on q in Theorem 1 (which is unfortunately outrageous). Indeed it is not clear why there should be any dependence at all and it would be nice to eliminate it if possible.

Organization. We give an overview of the proof of Theorem 1 in Section 2, where we also introduce the main technical theorem of this paper (Theorem 2). We also describe our technical contributions in this section, contrasting the current proof with those of [7,13], which we modify. In Section 3 we give examples of family of lifted codes for which our main theorem applies. Some of the details are omitted from this version due to space considerations. A full version of this paper is available as [12].

2 Overview of Proof

2.1 Some Natural Tests

Our proof of Theorem 1 follows the paradigm used in [7] and [13]. Both works consider a natural family of tests (and not just the "most" natural test), and analyze their performance by studying the behavior of functions when restricted to "hyperplanes". We introduce the family of tests first.

From now onwards all codes we consider will be linear and affine-invariant unless we explicitly say otherwise. Given a base code $\mathcal{B} \subseteq \{\mathbb{F}_q^t \to \mathbb{F}_q\}$ and $n \geq \ell \geq t$, we let $\mathcal{L}_\ell = \mathrm{Lift}_\ell(\mathcal{B})$, with $\mathcal{F} = \mathcal{L}_n$. The *$\ell$-dimensional test* for membership in $\mathcal{F}$ works as follows: Pick a random ℓ-dimensional affine subspace A in $\mathbb{F}_q^n$ and accept f if and only if $f|_A \in \mathcal{L}_\ell$.

Let $\mathrm{Rej}_\ell(f)$ denote the probability with which the ℓ-dimensional test rejects. Our main theorem aims to show that $\mathrm{Rej}_\ell(f) = \Omega(\delta_{\mathcal{F}}(f))$ when $\ell = t$. As in previous works, our analysis will first lower bound $\mathrm{Rej}_\ell(f)$ for $\ell = t + O(1)$ and then relate the performance of this test to the performance of the t-dimensional test.

2.2 Overview of Proof of Main Theorem 1

The analysis of the performance of the ℓ-dimensional tests is by induction on the number of variables n and based on the behaviour of functions when restricted to "hyperplanes". A *hyperplane* in $\mathbb{F}_q^n$ is an affine subspace of dimension $n-1$. In many future calculations it will be useful to know the number of hyperplanes in $\mathbb{F}_q^n$. We note that this number is $q^n + q^{n-1} + \cdots + 1 = q^n(1 + o(1))$.

The inductive strategy to analyzing $\mathrm{Rej}_\ell(f)$ is based on the observation that $\mathrm{Rej}_\ell(f) = \mathbb{E}_H[\mathrm{Rej}_\ell(f|_H)]$ where H is a uniform hyperplane. If we know that on most hyperplanes $\delta_{\mathcal{L}_{n-1}}(f|_H)$ is large, then we can prove the right hand side above is large by induction. Thus the inductive strategy relies crucially on

showing that if f is far from $\mathcal{F}$, then $f|_H$ can not be too close to $\mathcal{L}_{n-1}$ on too many hyperplanes. We state this technical result in the contrapositive form below.

Theorem 2 (Main technical). *For every q there exists $\tau < \infty$ such that the following holds: Let $\mathcal{B} \subseteq \{\mathbb{F}_q^t \to \mathbb{F}_q\}$ be an affine-invariant linear code and for $\ell \geq t$ let $\mathcal{L}_\ell = \mathrm{Lift}_\ell(\mathcal{B})$. For $n > t$, let $f : \mathbb{F}_q^n \to \mathbb{F}_q$ be a function and $H_1, \ldots, H_k$ be hyperplanes in $\mathbb{F}_q^n$ such that $\delta_{\mathcal{L}_{n-1}}(f|_{H_i}) \leq \delta$ for every $i \in [k]$ for $\delta < \frac{1}{2}q^{-(t+1)}$. Then, if $k \geq q^{t+\tau}$, we have $\delta_{\mathcal{L}_n}(f) \leq 2\delta + 4(q-1)/k$.*

The theorem thus states that if f is sufficiently close to a lift of $\mathcal{B}$ on a sufficiently large number of hyperplanes, yet a very small number (independent of n) of hyperplanes, then f is close to a lift of $\mathcal{B}$. The dependence of the number of hyperplanes on q and t is actually important to our (and previous) analysis. The fact that it is some fixed multiple of q^t, where the multiple depends only on q and not on t, is crucial to the resulting performance.

Going from Theorem 2 above to Theorem 1 is relatively straightforward. In particular using Theorem 2 we can get a lower bound on $\mathrm{Rej}_{t+\tau}(f)$ without any changes to the proof of [13]. However going from such an analysis to a lower bound on $\mathrm{Rej}_t(f)$ involves some extra work, with complications similar to (but simpler than), those in the proof of Theorem 2 so we omit a discussion here.

The main contribution of this paper is the proof of Theorem 2. Here, the previous proofs, both in [7] and [13] crucially relied on properties of polynomials and in particular the first step in both proofs, when testing degree d polynomials, is to consider the case of f being a degree $d+1$ (or a degree $d+q$) polynomial. In our case there is no obvious candidate for the notion of a degree $d+1$ polynomial and it is abstracting such properties that forms the bulk of our work. In what follows we give an overview of some of the issues arising in such steps and how we deal with them.

2.3 Overview of Proof of Theorem 2

To understand our proof of Theorem 2 we need to give some background, specifically to the proofs from the previous work of [13]. Recall the analogous statement in [13] attempted to show that if f was far from being a polynomial of degree d, then the number of hyperplanes where f turns out to be close to being a degree d polynomial is at most $O(q^t)$ (where $t \approx d/q$, the exact number will not be important to us). [13] reasoned about this in a sequence of steps: (1) They first showed that any function of degree greater than d, stays of degree greater than d on at least $1/q$ fraction of all hyperplanes (provided $n > t$). (2) Next they reasoned about functions of degree $d+1$ and showed that such a function reduces its degree on at most $O(q^t)$ hyperplanes. (3) In the third step they consider a general function f that is *far* from being of degree d and show that the number of hyperplanes on which f becomes a degree d polynomial *exactly* is $O(q^t)$. (This is the step where the big-Oh becomes a really big-Oh.) (4) Finally, they show that for functions of the type considered in the previous step the number

of hyperplanes where they even get *close* to being of degree d is at most $O(q^t)$, thus yielding the analog of Theorem 2.

In implementing the program above (which is what we will end up doing) in our more general/abstract setting, our first bottleneck is that, for instance in Step (2) above, we don't have a notion of degree $d+1$ or some notion of functions that are "just outside our good set $\mathcal{F}$". Natural notions of things outside our set do exist, but they don't necessarily satisfy our needs. To understand this issue better, let us see why polynomials of degree $d + O(1)$ appear in the analysis of a theorem such as Theorem 2. Consider a simple case where $H_1, \ldots, H_q$ are parallel hyperplanes completely covering $\mathbb{F}_q^n$ and $\delta = 0$ so f is known to be a good function (member of $\mathcal{F}$, or degree d) when restricted to these hyperplanes. So, in the setting of testing polynomials of degree at most d, the hypothesis asserts that f restricted to these hyperplanes is a polynomial of degree at most d. For notational simplicity we assume that H_i is the hyperplane given by $x_1 = \eta_i$ where $\mathbb{F}_q = \{\eta_1, \ldots, \eta_q\}$. Then $f|_{H_i} = P_i(x_2, \ldots, x_n)$ for some polynomial P_i of degree d. By polynomial interpolation, it follows that f can be described as a degree $d+q-1$ polynomial in $x_1, \ldots, x_n$. The bulk of the analysis in [7,13] now attempts to use the remaining $K - q$ hyperplanes on which f reduces to degree at most d, in conjunction with the fact that f is a polynomial of degree at most $d + q - 1$ to argue that f is of degree at most d.

For us, the main challenge is that in the generic setting of the lift of some code $\mathcal{B}$, we don't have a ready notion of a degree $d + q - 1$ polynomial and so we have to define one. Thus the first step in this work is to define such a code. For our current discussion it suffices to say that there is an affine-invariant linear code, which we denote $\mathcal{F}^+$, which contains all "interpolating functions" of elements of $\mathcal{F}$ (so $\mathcal{F}^+$ contains every function f for which there exist some q parallel hyperplanes $H_1, \ldots, H_q$ such that $f|_{H_i}$ is a function in $\mathcal{L}_{n-1}$ for all i). Of course such a set is not useful if it does not have some nice structure. The key property of our definition of $\mathcal{F}^+$ is that it is the lift of a non-trivial code on at most $t+q-1$ dimensions. This definition of $\mathcal{F}^+$ and its analysis rely centrally on some of the structural understanding of affine-invariant linear codes derived in previous works [16,9,10,6,5,4]. Our analysis shows that $\mathcal{F}^+$ is almost as nice as $\mathcal{F}$, roughly analogous to the way the set of degree $d + q - 1$ polynomials is almost as nice as the set of degree d polynomials.

The notion of $\mathcal{F}^+$ turns out to be easy enough to use to be able to carry out the steps (3) and (4) in the program above by directly mimicking the proofs of [13], assuming Steps (1) and (2) hold. But Steps (1) and (2) turn out to be more tricky. So we turn to these, and in particular Step (2) next.

Our next barrier in extending the proofs of [13] is a notion of "canonical monomials" which play a crucial role in Step (2) of [13]. For a function of degree $d + 1$, the canonical monomial is a monomial of degree $d + 1$ supported on very few variables. The fact that the number of variables in the support is small, while the monomial remains a "forbidden one" turns out to be central to their analysis and allows them to convert questions of the form: "Does f become a polynomial of smaller degree on the hyperplane H?"(which are typically not

well-understood) to questions of the form "Does g become the zero polynomial when restricted to H?" (which is a very well-studied question).

In our case, we need to work with some function f in $\mathcal{F}^+$ which is not a function of $\mathcal{F}$. The fact that $\mathcal{F}^+$ is a lift of "few-dimensional" code, in principle ought to help us find a monomial supported on few variables that is not in $\mathcal{F}$. But isolating the "right one" to work with for f turns out to be a subtle issue and we work hard, and come up with a definition that is very specific to each function $f \in \mathcal{F}^+ \setminus \mathcal{F}$. (In contrast the canonical monomials of [13] were of similar structure for every function f.) Armed with this definition and some careful analysis we are able to simulate Step (2) in the program above. We give a few more details into this step below. Full details may be found in the full version of this paper.

Let $t^+ = t + q - 1$, and let $\mathcal{B}^+$ be a family on t^+ variables such that $\mathcal{F}^+$ is a lift of $\mathcal{B}^+$. Let $f \in \mathcal{F}^+ \setminus \mathcal{F}$. We first show that for every such f there exists an invertible affine transformation T and monomial $M \notin \mathcal{F}$ supported on the first t^+ variables such that $f \circ T$ is supported on M. We further assume that T is such that the degree of M is maximal. Without loss of generality we may assume T is just the identity transformation and so f is supported on M. Next we partition the space of all possible hyperplanes into $q^{t^+ + 1}$ sets (based on their coefficients on the first t^+ variables). Our goal is to show that in each set in the partition there are at most some constant (depending on q) number of hyperplanes such that f restricted to that hyperplane becomes a member of $\mathcal{F}$. To do so we extract from f a non-zero low-degree function g. (this function g depends on M and the set in the partition under consideration). We show that for the correct definition of g, it is the case that $f|_H \in \mathcal{F}$ only if $g|_H \equiv 0$. This brings us to the final task: to bound the number of hyperplanes on which $g|_H$ can be identically zero. For this part we show a simple lemma (see Lemma 4.8 in the full version) that shows that a low-degree function can only be zero on a small number of hyperplanes (bounded by a function of q and the degree, but independent of n). Putting the above ingredients together gives us a bound (of desired quality) on the number of hyperplanes H for which $f|_H \in \mathcal{F}$.

Finally, Step (1) is also dealt with similarly, using some of the same style of ideas as in the proof of Step (2).

3 New Testable Codes

In this section, we give some examples of codes with "nice" parameters that are testable with absolute soundness based on our main theorem (Theorem 1).

The need for such codes is motivated by the work of Barak et al. [3]. Their work used appropriate Reed-Muller codes over $\mathbb{F}_2$. Our work gives the second family of codes that is known to satisfy their requirements. We point out that Guo et al. [11] also give codes motivated by the work of [3], but their codes are not, thus far, known to be testable with absolute soundness and so fail to meet all the requirements of [3]. Our codes fall within the class of "lifted" codes studied by [11], but were not analyzed there. Here we use analysis similar to

their to analyze the rate and distance of our codes, while the testing follows from our main theorem.

The code. Our codes are defined by three parameters: a real number $\epsilon > 0$ and two integers s and n. The code $\mathcal{F} = \mathcal{F}_{\epsilon,s,n}$ is obtained as follows: Let $q = 2^s$, and let $\ell = \lfloor \frac{1}{s} \log 1/\epsilon \rfloor$. Let $\mathcal{B} = \{f : \mathbb{F}_q^{n-\ell} \to \mathbb{F}_2 | \sum_{\boldsymbol{x} \in \mathbb{F}_q^{n-\ell}} f(\boldsymbol{x}) = 0\}$. Let $\mathcal{F} = \mathrm{Lift}_n(\mathcal{B})$.

Basic parameters:

Proposition 2. *For every ϵ, s and n the code $\mathcal{F} = \mathcal{F}_{\epsilon,s,n}$ has block length $N = 2^{sn}$, (absolute, non-normalized) distance at least $1/\epsilon$ and dimension at least $2^{sn} - \left(\binom{n}{\ell}^s + \sum_{i=0}^{s\ell-1} \binom{ns}{i}\right)$.*

Proof. The size of the block length can be easily verified and the distance follows from Lemmas 3.11. and 3.12. in Guo et. al. [11] analyzed the dimension of the code $\mathcal{F}_{\epsilon,s,n}$ for the case in which $s = \log(1/\epsilon)$ (so $\ell = 1$). More specifically, given a degree pattern $a = (a_1, \ldots, a_n)$ with $\{a_i\}_{i=1}^n \subseteq \mathbb{Z}_q$, let $a_i^{(j)}$ denote the j-th bit of the binary expansion of a_i. Let $M(a)$ denote the $n \times s$ matrix with entries $M(a)_{i,j} = a_i^{(j)}$. Guo et. al. show that in the special case in which $\ell = 1$ the code $\mathcal{F}_{\epsilon,s,n}$ contains in its support all monomials with degree pattern $a = (a_1, \ldots, a_n)$ such that there exists a column in $M(a)$ with at least two zeroes. This readily implies a bound of $2^{sn} - (n+1)^\ell$ on the dimension of their code.

A similar analysis shows that our code $\mathcal{F}_{\epsilon,s,n}$ contains all monomials with degree pattern $a = (a_1, \ldots, a_n)$ where the matrix $M(a)$ has at least $s\ell + 1$ zeroes, or the matrix has $s\ell$ zeroes and there exists a column in $M(a)$ with at least $\ell + 1$ zeros. The lower bound on the dimension follows.

Testability. The following is an immediate application of Theorem 1.

Proposition 3. *For every s there exists a constant $\tau > 0$ such that for every ϵ and n the code $\mathcal{F} = \mathcal{F}_{\epsilon,s,n}$ is testable by a test that makes ϵN queries, accepts codewords with probability one, while rejecting all functions $f : \mathbb{F}_q^n \to \mathbb{F}_2$ with probability at least $\tau \cdot \delta(f, \mathcal{F})$.*

We remark that the dimension of our codes, for any choice of N and ϵ is strictly better than that of the codes used in [3] which have dimension $2^{sn} - \sum_{i=0}^{s\ell} \binom{sn}{i} \approx 2^{sn} - \frac{1}{\sqrt{2\pi s\ell}}(en/\ell)^{s\ell}$. An important parameter for them is the "co-dimension" of their code (block length minus the dimension, or the dimension of the dual code), which thus turns out to be roughly $\frac{1}{\sqrt{2\pi s\ell}}(en/\ell)^{s\ell}$ from the above expression. (A smaller codimension is better for their application.) Simplifying the dimension of our code from Proposition 2, we see that the codimension of our code is smaller by a multiplicative factor of roughly $O(\ell^{s/2-1})$, making our codes noticeably better. Unfortunately such changes do not alter the essential relationship between $N = 2^{sn}$, the parameter ϵ (which determines the locality of the tester) and the codimension of the code. The following theorem summarizes the performance of our codes.

Theorem 3. *For every positive s there exists a constant τ such that for every sufficiently small ϵ and sufficiently large N there exists a code of block length N, codimension $\left(\log \frac{1}{\epsilon}\right)^{-s} \cdot \left(\frac{e \log N}{\log \frac{1}{\epsilon}}\right)^{\log \frac{1}{\epsilon}}$ that is testable with a tester that makes $\epsilon \cdot N$ queries accepting codewords with probability one, while rejecting words at distance δ with probability at least $\tau \cdot \delta$.*

To contrast, the corresponding result in [3] would assert the existence of a positive constant s for which the above held.

References

1. Alon, N., Kaufman, T., Krivelevich, M., Litsyn, S., Ron, D.: Testing Reed-Muller codes. IEEE Transactions on Information Theory 51(11), 4032–4039 (2005)
2. Arora, S., Sudan, M.: Improved low degree testing and its applications. Combinatorica 23(3), 365–426 (2003)
3. Barak, B., Gopalan, P., Håstad, J., Meka, R., Raghavendra, P., Steurer, D.: Making the long code shorter, with applications to the unique games conjecture. In: FOCS. IEEE Computer Society (2012)
4. Ben-Sasson, E., Grigorescu, E., Maatouk, G., Shpilka, A., Sudan, M.: On sums of locally testable affine invariant properties. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) APPROX/RANDOM 2011. LNCS, vol. 6845, pp. 400–411. Springer, Heidelberg (2011)
5. Ben-Sasson, E., Maatouk, G., Shpilka, A., Sudan, M.: Symmetric LDPC codes are not necessarily locally testable. In: IEEE Conference on Computational Complexity, pp. 55–65. IEEE Computer Society (2011)
6. Ben-Sasson, E., Sudan, M.: Limits on the rate of locally testable affine-invariant codes. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) APPROX/RANDOM 2011. LNCS, vol. 6845, pp. 412–423. Springer, Heidelberg (2011)
7. Bhattacharyya, A., Kopparty, S., Schoenebeck, G., Sudan, M., Zuckerman, D.: Optimal testing of reed-muller codes. In: FOCS, pp. 488–497. IEEE Computer Society (2010)
8. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. In: STOC, pp. 73–83. ACM (1990)
9. Grigorescu, E., Kaufman, T., Sudan, M.: 2-transitivity is insufficient for local testability. In: IEEE Conference on Computational Complexity, pp. 259–267. IEEE Computer Society (2008)
10. Grigorescu, E., Kaufman, T., Sudan, M.: Succinct representation of codes with applications to testing. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX and RANDOM 2009. LNCS, vol. 5687, pp. 534–547. Springer, Heidelberg (2009)
11. Guo, A., Kopparty, S., Sudan, M.: New affine-invariant codes from lifting. In: Proceedings of ITCS 2013 (to appear, 2013)
12. Haramaty, E., Ron-Zewi, N., Sudan, M.: Absolutely sound testing of lifted codes. Electronic Colloquium on Computational Complexity (ECCC) 20, 30 (2013)
13. Haramaty, E., Shpilka, A., Sudan, M.: Optimal testing of multivariate polynomials over small prime fields. In: FOCS, pp. 629–637. IEEE Computer Society (2011)
14. Jutla, C.S., Patthak, A.C., Rudra, A., Zuckerman, D.: Testing low-degree polynomials over prime fields. Random Struct. Algorithms 35(2), 163–193 (2009)

15. Kaufman, T., Ron, D.: Testing polynomials over general fields. SIAM Journal of Computing 36(3), 779–802 (2006)
16. Kaufman, T., Sudan, M.: Algebraic property testing: The role of invariance. Electronic Colloquium on Computational Complexity (ECCC) 14(111) (2007)
17. Kaufman, T., Sudan, M.: Algebraic property testing: the role of invariance. In: STOC, pp. 403–412. ACM (2008)
18. Raz, R., Safra, S.: A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In: STOC, pp. 475–484. ACM (1997)
19. Ron-Zewi, N., Sudan, M.: A new upper bound on the query complexity for testing generalized reed-muller codes. In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) APPROX/RANDOM 2012. LNCS, vol. 7408, pp. 639–650. Springer, Heidelberg (2012)
20. Rubinfeld, R., Sudan, M.: Robust characterizations of polynomials with applications to program testing. SIAM J. on Computing 25(2), 252–271 (1996)

On the Average Sensitivity and Density of k-CNF Formulas*

Dominik Scheder[1] and Li-Yang Tan[2,**]

[1] Aarhus University
[2] Columbia University

Abstract. We study the relationship between the average sensitivity and density of k-CNF formulas via the isoperimetric function $\varphi : [0,1] \to \mathbb{R}$,

$$\varphi(\mu) = \max\left\{ \frac{\mathbf{AS}(F)}{\mathbf{CNF\text{-}width}(F)} : \mathbf{E}[F(\boldsymbol{x})] = \mu \right\},$$

where the maximum is taken over all Boolean functions $F : \{0,1\}^* \to \{0,1\}$ over a finite number of variables and $\mathbf{AS}(F)$ is the average sensitivity of F. Building on the work of Boppana [1] and Traxler [2], and answering an open problem of O'Donnell, Amano [3] recently proved that $\varphi(\mu) \leq 1$ for all $\mu \in [0,1]$. In this paper we determine φ exactly, giving matching upper and lower bounds. The heart of our upper bound is the Paturi-Pudlák-Zane (PPZ) algorithm for k-SAT [4], which we use in a unified proof that sharpens the three incomparable bounds of Boppana, Traxler, and Amano.

We extend our techniques to determine φ when the maximum is taken over monotone Boolean functions F, further demonstrating the utility of the PPZ algorithm in isoperimetric problems of this nature. As an application we show that this yields the largest known separation between the average and maximum sensitivity of monotone Boolean functions, making progress on a conjecture of Servedio.

Finally, we give an elementary proof that $\mathbf{AS}(F) \leq \log(s)(1 + o(1))$ for functions F computed by an s-clause CNF, which is tight up to lower order terms. This sharpens and simplifies Boppana's bound of $O(\log s)$ obtained using Håstad's switching lemma.

1 Introduction

The average sensitivity of a Boolean function $F : \{0,1\}^n \to \{0,1\}$ is a fundamental and well-studied complexity measure. The sensitivity of F at an input $x \in \{0,1\}^n$, denoted $S(F,x)$, is the number of coordinates $i \in [n]$ of x such

* The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which this work was performed.

** Part of this research was completed while visiting KTH Royal Institute of Technology, partially supported by ERC Advanced Investigator Grant 226203.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 683–698, 2013.

that $F(x) \neq F(x \oplus e_i)$, where $x \oplus e_i$ denotes x with its i-th coordinate flipped. The average sensitivity of F, denoted $\mathbf{AS}(F)$, is the expected number of sensitive coordinates of F at an input $\boldsymbol{x}$ drawn uniformly at random from $\{0,1\}^n$. Viewing F as the indicator of a subset $A_F \subseteq \{0,1\}^n$, the average sensitivity of F is proportional to the number of edges going from A_F to its complement, and so $\mathbf{AS}(F)$ may be equivalently viewed as a measure of the normalized edge boundary of A_F.

The average sensitivity of Boolean functions was first studied in the computer science literature by Ben-Or and Linial [5] in the context of distributed computing. Owing in part to connections with the Fourier spectrum of F established in the celebrated work of Kahn, Kalai, and Linial [6], this complexity measure has seen utility throughout theoretical computer science, receiving significant attention in a number of areas spanning circuit complexity [7,8,9][1], learning theory [11,12,13], random graphs [14,15,16], social choice theory, hardness of approximation [17], quantum query complexity [18], property testing [19], *etc.* We remark that the study of average sensitivity in combinatorics predates its introduction in computer science. For example, the well-known edge-isoperimetric inequality for the Hamming cube [20,21,22,23] yields tight extremal bounds on the average sensitivity of Boolean functions in terms of the number of its satisfying assignments.

The focus of this paper is on the average sensitivity of k-CNF formulas, the AND of ORs of k or fewer variables; by Boolean duality our results apply to k-DNF formulas as well. Upper bounds on the average sensitivity of small-depth AC^0 circuits are by now classical results, having been the subject of study in several early papers in circuit complexity [7,24,1,25]. Despite its apparent simplicity, though, gaps remain even in our understanding of the average sensitivity of depth-2 AC^0 circuits. The starting point of this research was the following basic question:

Question 1. What is the maximum average sensitivity of a k-CNF formula $F : \{0,1\}^n \to \{0,1\}$ that is satisfied by a μ fraction of assignments?

An easy folkloric argument (first appearing explicitly in [1]) gives an upper bound of $2(1-\mu)k$. The maximum of $2k$ attained by this bound is a multiplicative factor of 2 away from the lower bound of k witnessed by the parity function over k variables, leading O'Donnell to ask if there is indeed a matching upper bound of k [26]. O'Donnell's question was answered in a sequence of works by Traxler [2] and Amano [3], with Traxler proving a bound of $2\mu \log_2(1/\mu)k$ (attaining a maximum of $\sim 1.062k$ at $\mu = 1/e$), followed by Amano's bound of k independent of μ. These three incomparable bounds are shown in Figure 1 where they are normalized by k.

The natural question at this point is: what is the true dependence on μ? In this work we answer this question by giving matching upper and lower bounds. Traxler's upper bound of $2\mu \log_2(1/\mu)k$ is easily seen to be tight at the points

[1] Though couched in different terminology, Khrapchenko's classical lower bound [10] on the formula size of Boolean functions also relies implicitly on average sensitivity.

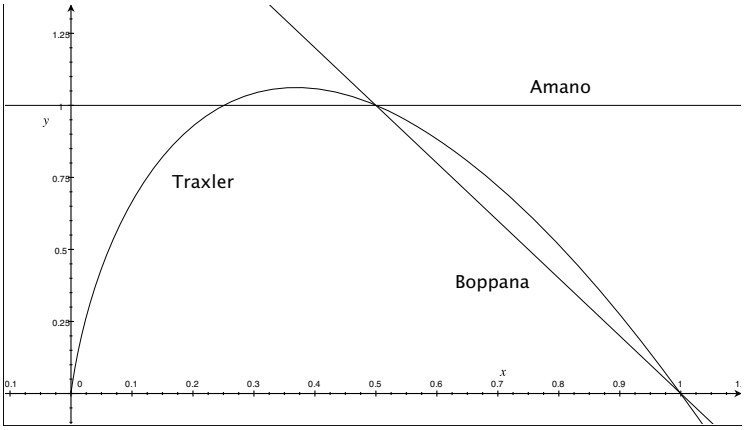


Fig. 1. The upper bounds of Boppana, Traxler, and Amano, normalized by k

$\mu = 2^{-I}$ for all positive integers $I \in \mathbb{N}$, since the AND of I variables is a 1-CNF with average sensitivity $2\mu \log_2(1/\mu)$, but we are not aware of any other matching lower bounds prior to this work. Like Traxler and Amano, the main technical tool for our upper bound is the Paturi-Pudlák-Zane (PPZ) randomized algorithm for k-SAT. We remark that this is not the first time the PPZ algorithm has seen utility beyond the satisfiability problem; in their original paper the authors use the algorithm and its analysis to obtain sharp lower bounds on the size of depth-3 AC^0 circuits computing the parity function.

We extend our techniques to determine φ when the maximum is taken over monotone Boolean functions F, further demonstrating the utility of the PPZ algorithm in isoperimetric problems of this nature. As an application we show that this yields the largest known separation between the average and maximum sensitivity of monotone functions, making progress on a conjecture of Servedio. Finally, we give an elementary proof that $\mathbf{AS}(F) \leq \log(s)(1+o(1))$ for functions F computed by an s-clause CNF; such a bound that is tight up to lower order terms does not appear to have been known prior to our work.

1.1 Our Results

Our main object of study is the following isoperimetric function:

Definition 1. *Let* $\varphi : [0,1] \to \mathbb{R}$ *be the function:*

$$\varphi(\mu) = \max\left\{ \frac{\mathbf{AS}(F)}{\mathbf{CNF\text{-}width}(F)} : \mathbf{E}[F(\boldsymbol{x})] = \mu \right\},$$

where the maximum is taken over all Boolean functions $F : \{0,1\}^* \to \{0,1\}$ *over a finite number of variables.*

Note that $\mathbf{E}[F(\boldsymbol{x})] = a2^{-b}$ for $a, b \in \mathbb{N}$, and thus $\varphi(\mu)$ is well-defined only at those points. However, these points are dense within the interval $[0,1]$ and

thus one can continuously extend φ to all of $[0,1]$. As depicted in Figure 1 the upper bounds of Boppana, Traxler, and Amano imply that $\varphi(\mu) \leq \min\{2(1-\mu), 2\mu\log(1/\mu), 1\}$. In this paper we determine φ exactly, giving matching upper and lower bounds.

Theorem 1. *$\varphi(\mu) : [0,1] \to \mathbb{R}$ is the piecewise linear continuous function that evaluates to $2\mu\log_2(1/\mu)$ when $\mu = 2^{-I}$ for some $I \in \mathbb{N} = \{0,1,2,\ldots\}$, and is linear between these points[2]. That is, if $\mu = t \cdot 2^{-(I+1)} + (1-t)\cdot 2^I$ for some $I \in \mathbb{N}$ and $t \in [0,1]$, then*

$$\varphi(\mu) = t \cdot \frac{(I+1)}{2^I} + (1-t)\cdot\frac{I}{2^{I-1}}.$$

We extend our techniques to also determine the variant of φ where the maximum is taken only over monotone Boolean functions. The reader familiar with the PPZ algorithm will perhaps recall the importance of Jensen's inequality in its analysis. Jensen's inequality is very helpful for dealing with random variables whose correlations one does not understand. It turns out that in case of monotone CNF formulas, certain events are *positively correlated* and we can replace Jensen's inequality by the FKG inequality [27], leading to a substantial improvement in the analysis.

Theorem 2 (Upper bound for monotone k-CNFs). *Let F be a monotone k-CNF formula and $\mu = \mathbf{E}[f(\boldsymbol{x})]$. Then $\mathbf{AS}(F) \leq 2k\mu\ln(1/\mu)(1+\varepsilon_k)$ for some ε_k that goes to 0 as k grows.*[3]

Theorem 3 (Lower bound for monotone k-CNFs). *Let $\mu \in [0,1]$ and $k \in \mathbb{N}$. There exists a monotone k-CNF formula F with $\mathbf{E}[F(\boldsymbol{x})] = \mu \pm \varepsilon_k$ and $\mathbf{AS}(F) \geq 2k\mu\ln(1/\mu)(1-\varepsilon_k)$ for some ε_k that goes to 0 as k grows.*

We apply Theorem 2 to obtain a separation between the average and maximum sensitivity of monotone Boolean functions, making progress on a conjecture of Servedio [28]. Our result improves on the current best gap of $\mathbf{AS}(f) \leq \sqrt{2/\pi}\cdot\mathbf{S}(f)(1+o(1)) \approx 0.797\cdot\mathbf{S}(f)(1+o(1))$, which follows as a corollary of an isoperimetric inequality of Blais [29].

Corollary 1. *Let f be a monotone Boolean function. Then $\mathbf{AS}(F) \leq \ln(2)\cdot\mathbf{S}(F)(1+o(1)) \leq 0.694\cdot\mathbf{S}(f)(1+o(1))$, where $o(1)$ is a term that goes to 0 as $\mathbf{S}(F)$ grows.*

Finally, we give an elementary proof that $\mathbf{AS}(F) \leq \log(s)(1+o(1))$ for functions F computed by an s-clause CNF, which is tight up to lower order terms by considering the parity of $\log s$ variables. This sharpens and simplifies Boppana's bound of $O(\log s)$ obtained using Håstad's switching lemma.

Theorem 4. *Let F be an s-clause CNF. Then $\mathbf{AS}(F) \leq \log s + \log\log s + O(1)$.*

[2] We use the fact that $0\log_2(1/0) = 0$ here.

[3] Note that the additive term of ε_k is necessary since $\mathbf{AS}(F) = 1$ when $F = x_1$.

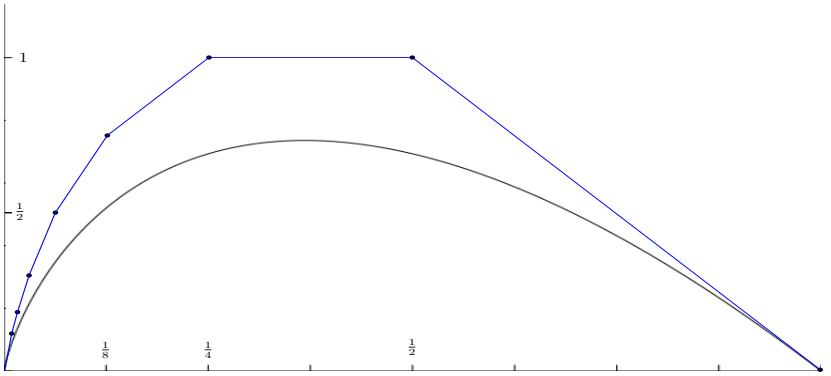


Fig. 2. Our matching bounds for all functions (top), and for monotone functions (bottom)

1.2 Preliminaries

Throughout this paper all probabilities and expectations are with respect to the uniform distribution, and logarithms are in base 2 unless otherwise stated. We adopt the convention that the natural numbers $\mathbb{N}$ include 0. We use boldface letters (*e.g.* $\boldsymbol{x}$, $\boldsymbol{\pi}$) to denote random variables.

For any Boolean function $F : \{0,1\}^n \to \{0,1\}$, we write $\mu(F) \in [0,1]$ to denote the density $\mathbf{E}_{x\in\{0,1\}^n}[F(\boldsymbol{x})]$ of F, and $\mathrm{sat}(F) \subseteq \{0,1\}^n$ to denote the set of satisfying assignments of F (and so $|\mathrm{sat}(F)| = \mu \cdot 2^n$). The CNF width of F, which we will denote $\mathbf{CNF\text{-}width}(F)$, is defined to be the smallest $k \in [n]$ such that F is computed by a k-CNF formula; similarly, $\mathbf{DNF\text{-}width}(F)$ is the smallest k such that F is computed by a k-DNF formula. Note that by Boolean duality, we have the relation $\mathbf{CNF\text{-}width}(F) = \mathbf{DNF\text{-}width}(\neg F)$.

Definition 2. *Let $F : \{0,1\}^n \to \{0,1\}$ and $x \in \{0,1\}^n$. For any $i \in [n]$, we say that F is sensitive at coordinate i on x if $F(x) \neq F(x \oplus e_i)$, where $x \oplus e_i$ denotes x with its i-th coordinate flipped, and write $S(F,x,i)$ as the indicator for this event. The sensitivity of F at x, denoted $S(F,x)$, is $\#\{i \in [n] \colon F(x) \neq F(x \oplus e_i)\} = \sum_{i=1}^n S(F,x,i)$. The average sensitivity and maximum sensitivity of F, denoted $\mathbf{AS}(F)$ and $\mathbf{S}(F)$ respectively, are defined as follows:*

$$\mathbf{AS}(F) = \mathop{\mathbf{E}}_{x\in\{0,1\}^n}[S(F,\boldsymbol{x})], \qquad \mathbf{S}(F) = \max_{x\in\{0,1\}^n}[S(F,x)].$$

We will need the following basic fact:

Fact 11. *Let $F : \{0,1\}^n \to \{0,1\}$ and $\mu = \mathbf{E}[F(\boldsymbol{x})]$. Then $\mathbf{E}_{x\in\mathrm{sat}(F)}[S(F,\boldsymbol{x})] = \mathbf{AS}(F)/2\mu$.*

Proof. This follows by noting that

$$\mathbf{AS}(f) = \mathop{\mathbf{E}}_{x\in\{0,1\}^n}[S(F,\boldsymbol{x})] = \mathop{\mathbf{E}}_{x\in\{0,1\}^n}[2{\cdot}S(F,\boldsymbol{x}){\cdot}\mathbf{1}_{[F(\boldsymbol{x})=1]}] = 2\mu \mathop{\mathbf{E}}_{x\in\mathrm{sat}(F)}[S(F,\boldsymbol{x})].$$

Here the second identity holds by observing that for any $x \in \mathrm{sat}(F)$ and coordinate $i \in [n]$ on which F is sensitive at x, we have that $x \oplus e_i \notin \mathrm{sat}(F)$ and F is sensitive on i at $x \oplus e_i$. □

We remark that Boppana's bound follows easily from 11 and Boolean duality. For any k-CNF F with density $\mathbf{E}[f(\boldsymbol{x})] = \mu$, its negation $\neg F$ is a k-DNF with density $(1-\mu)$ and $\mathbf{AS}(\neg F) = \mathbf{AS}(F)$. Applying Fact 11 to $\neg F$ and noting that every satisfying assignment of a k-DNF has sensitivity at most k, we conclude that $\mathbf{AS}(F) = \mathbf{AS}(\neg F) \leq 2(1-\mu)k$.

2 The PPZ Algorithm

The main technical tool for our upper bounds in both Theorems 1 and 2 is the PPZ algorithm (Figure 3), a remarkably simple and elegant randomized algorithm for k-SAT discovered by and named after Paturi, Pudlák, and Zane [4]. Perhaps somewhat surprisingly, the utility of the PPZ algorithm extends beyond its central role in the satisfiability problem. Suppose the PPZ algorithm is run on a k-CNF F, for which it is searching for an satisfying assignment $x \in \mathrm{sat}(F)$. Since the algorithm is randomized and may not return a satisfying assignment, it defines a probability distribution on $\mathrm{sat}(F) \cup \{\texttt{failure}\}$.

The key observation underlying the analysis of PPZ is that a satisfying assignment x for which $S(F,x)$ is large receives a higher probability under this distribution than its less sensitive brethren; the exact relationship depends on $\mathbf{CNF\text{-}width}(F)$, and is made precise by the Satisfiability Coding Lemma (Lemma 2). Since the probabilities of the assignments sum to at most 1, it follows that there cannot be too many high-sensitivity assignments. This intuition is the crux of the sharp lower bounds of Paturi, Pudlák, and Zane on the size of depth-3 AC^0 circuits computing parity; it is also the heart of Traxler's, Amano's, and our upper bounds on the average sensitivity of k-CNF formulas.

Let us add some bookkeeping to this algorithm. For every satisfying assignment $x \in \mathrm{sat}(F)$, permutation $\pi : [n] \to [n]$, and coordinate $i \in [n]$, we introduce an indicator variable $T_i(x, \pi, F)$ that takes value 1 iff the assignment x_i to the

The ppz algorithm takes as input a k-CNF formula F and a permutation $\pi : [n] \to [n]$.

1. for $i = 1$ to n:
2. if $x_{\pi(i)}$ occurs in a unit clause in F then set $x_{\pi(i)} \leftarrow 1$ in F.
3. else if $\overline{x}_{\pi(i)}$ occurs in a unit clause in F then set $x_{\pi(i)} \leftarrow 0$ in F.
4. else toss a fair coin and set $x_{\pi(i)}$ to 0 or 1 uniformly at random.

If $F \equiv 1$, the algorithm has found a satisfying assignment and returns it. Otherwise the algorithm reports `failure`.

Fig. 3. The PPZ k-SAT algorithm

i-th coordinate was decided by a coin *T*oss, conditioned on PPZ returning x on inputs F and π (which we denote as $\mathtt{ppz}(F,\pi)=x$). We also introduce the dual indicator variable $I_i(x,\pi,F)=1-T_i(x,\pi,F)$, which takes value 1 if the the assignment x_i was *I*nferred. We define $T(x,\pi,F)=T_1(x,\pi,F)+\cdots+T_n(x,\pi,F)$ to be the total number of coin tosses, and similarly $I(x,\pi,F)=I_1(x,\pi,F)+\ldots+I_n(x,\pi,F)=n-T(x,\pi,F)$ to be the number of inference steps. Note that if $x\in\mathrm{sat}(F)$ and we condition on the event $\mathtt{ppz}(F,\pi)=x$, then all coin tosses of the algorithm are determined and $T=T(x,\pi,F)$ becomes some constant in $\{0,1,\ldots,n\}$; likewise for $I=I(x,\pi,F)$. The next lemma follows immediately from these definitions:

Lemma 1 (Probability of a solution under PPZ [4]). *Let F be a CNF formula over n variables and $x\in\mathrm{sat}(F)$. Let π be a permutation over the variables. Then*

$$\mathbf{Pr}[\mathtt{ppz}(F,\pi)=x]=2^{-T(x,\pi,F)}=2^{-n+I(x,\pi,F)}\ , \tag{1}$$

where $T(x,\pi,F)$ is the number of coin tosses used by the algorithm when finding x.

For completeness, we include a proof of the simple but crucial Satisfiability Coding Lemma:

Lemma 2 (Satisfiability Coding Lemma [4]). *Let F be a k-CNF formula and let $x\in\mathrm{sat}(F)$. If F is sensitive at coordinate i on x then $\mathbf{E}_\pi[I_i(x,\boldsymbol{\pi},F)]\geq 1/k$, and otherwise $I_i(x,\pi,F)=0$ for all permutations π. Consquently, by linearity of expectation $\mathbf{E}_\pi[I(x,\boldsymbol{\pi},F)]\geq S(F,x)/k$.*

Proof. Without loss of generality we assume that $x=(1,\ldots,1)$, and since we condition on $\mathtt{ppz}(F,\pi)=x$, all coin tosses made by the algorithm yield a 1. If F is sensitive to i at x, then certainly there must exist a clause C in which x_i is the only satisfied literal. That is, $C=x_i\vee\overline{x}_{i_2}\vee\cdots\vee\overline{x}_{i_k}$. With probabilitiy $1/k$, variable i comes after $i_2,\ldots,i_k$ in the permutation π and in this case, the PPZ algorithm has already set the variables $x_{i_2},\ldots,x_{i_k}$ to 1 when it processes x_i. Thus, F will contain the unit clause $\{x_i\}$ at this point, and PPZ will not toss a coin for x_i (*i.e.* the value of x_i is forced), which means that $I_i(x,\pi,F)=1$. Thus, $\mathbf{E}_\pi[I_i(x,\boldsymbol{\pi},F)]\geq 1/k$. On the other hand if F is not sensitive to i at x, then every clause containing x_i also contains a second satisfied literal. Thus, PPZ will never encounter a unit clause containing only x_i and therefore $I_i(x,\pi,F)=0$ for all permutations π. □

3 Average Sensitivity of k-CNFs: Proof of Theorem 1

3.1 The Upper Bound

Let $g:\mathbb{R}\to\mathbb{R}$ be any monotone increasing convex function such that $g(I)\leq 2^I$ for $I\in\mathbb{N}$. Choose a uniformly random permutation π of the variables of F and

run PPZ. Summing over all $x \in \mathrm{sat}(F)$ and applying Lemma 1, we first note that

$$\begin{aligned} 1 \;\geq\; \sum_{x \in \mathrm{sat}(F)} \mathbf{Pr}[\mathsf{ppz}(F,\boldsymbol{\pi}) = x] &= \sum_{x \in \mathrm{sat}(F)} \mathop{\mathbf{E}}_{\pi} \left[2^{-T(x,\boldsymbol{\pi},F)}\right] \\ &= 2^{-n} \sum_{x \in \mathrm{sat}(F)} \mathbf{E}_{\pi} \left[2^{I(x,\boldsymbol{\pi},F)}\right] \\ &= \mu \mathop{\mathbf{E}}_{x \in \mathrm{sat}(F),\pi} \left[2^{I(\boldsymbol{x},\boldsymbol{\pi},F)}\right]. \end{aligned} \tag{2}$$

Next by our assumptions on g, we have

$$\mu \mathop{\mathbf{E}}_{x \in \mathrm{sat}(F),\pi} \left[2^{I(\boldsymbol{x},\boldsymbol{\pi},F)}\right] \geq \mu \mathop{\mathbf{E}}_{x \in \mathrm{sat}(F),\pi} [g(I(\boldsymbol{x},\boldsymbol{\pi},F))] \geq \mu \cdot g\left(\mathop{\mathbf{E}}_{x \in \mathrm{sat}(F),\pi} [I(\boldsymbol{x},\boldsymbol{\pi},F)]\right),$$

where the first inequality holds since g satisfies $g(I) \leq 2^I$ for all $I \in \mathbb{N}$, and the second follows from Jensen's inequality and convexity of g. Combining these inequalities and applying the Satisfiability Coding Lemma (Lemma 2), we get

$$1 \geq \mu \cdot g\left(\mathop{\mathbf{E}}_{x \in \mathrm{sat}(F),\pi} [I(\boldsymbol{x},\boldsymbol{\pi},F)]\right) \geq \mu \cdot g\left(\mathop{\mathbf{E}}_{x \in \mathrm{sat}(F)} \left[\frac{S(F,\boldsymbol{x})}{k}\right]\right) = \mu \cdot g\left(\frac{\mathbf{AS}(F)}{2\mu k}\right).$$

Here we have used the assumption that g is monotone increasing in the second inequality, and Fact 11 for the final equality. Solving for $\mathbf{AS}(F)$, we obtain the following upper bound:

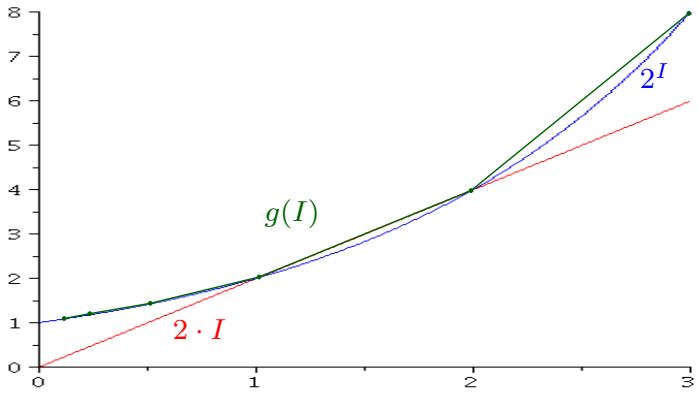
$$\mathbf{AS}(F) \leq 2\mu g^{-1}(1/\mu) \cdot k. \tag{3}$$

At this point we note that we can easily recover Traxler, Amano, and Boppana's bounds from Equation (3) simply by choosing the appropriate function $g : \mathbb{R} \to \mathbb{R}$ that satisfies the necessary conditions (*i.e.* monotone increasing, convex, and $g(I) \leq 2^I$ for all $I \in \mathbb{N}$).

- If $g(I) = 2^I$, then (3) becomes $\mathbf{AS}(F) \leq 2\mu \log(1/\mu) \cdot k$, which is Traxler's bound.
- If $g(I) = 2I$, we obtain $\mathbf{AS}(F) \leq k$, which is Amano's bound.
- If $g(I) = 1 + I$, we obtain Boppana's bound of $\mathbf{AS}(F) \leq 2(1-\mu) \cdot k$.[4]

We pick g to be the largest function that is monotone increasing, convex, and at most 2^I for all integers I. This is the convex envelope of the integer points $\big(I, 2^I\big)$:

[4] This observation was communicated to us by Lee [30].



That is, g is the continuous function such that $g(I) = 2^I$ whenever $I \in \mathbb{N}$, and is linear between these integer points. Thus, the function $g^{-1}(1/\mu)$ is piecewise linear in $1/\mu$, and $2\mu g^{-1}(1/\mu) \cdot k$ is piecewise linear in μ. Therefore we obtain an upper bound on $\varphi(\mu)$ that is $2\mu \log(1/\mu)$ if $\mu = 2^{-I}$ for $I \in \mathbb{N}$ and piecewise linear between these points. This proves the upper bound in Theorem 1.

3.2 The Lower Bound

We will need a small observation:

Lemma 3. *Let $k, \ell \in \mathbb{N}_0$, and set $\mu = 2^{-\ell}$. There exists a Boolean function $F : \{0,1\}^n \to \{0,1\}$ with* **CNF-width**$(F) = k$ *and* $\mathbf{AS}(F) = 2\mu \log(1/\mu) \cdot k$.

Proof. We introduce $k \cdot \ell$ variables $x_j^{(i)}$, $1 \le i \le \ell$, $1 \le j \le k$ and let F be (k,ℓ)-*block parity*, defined as

$$F := \bigwedge_{i=1}^{\ell} \bigoplus_{j=1}^{k} x_j^{(i)} \ .$$

Note that F has density $\mathbf{E}[F(\boldsymbol{x})] = 2^{-\ell} = \mu$, and every satisfying assignment has sensitivity exactly $k\ell$. Thus by Fact 11, $\mathbf{AS}(F) = 2\mu\, \mathbf{E}_{x \in \mathrm{sat}}[S(F, \boldsymbol{x})] = 2k\ell 2^{-\ell} = 2k\mu \log(1/\mu) \cdot k$. □

By Lemma 3, for every $k \in \mathbb{N}$ there is a k-CNF which is a tight example for our upper bound whenever $\mu = 2^{-\ell}$ and $\ell \in \mathbb{N}$. The main idea is to interpolate φ linearly between $\mu = 2^{-\ell-1}$ and $2^{-\ell}$ for all integers ℓ. If μ is not an integer power of $1/2$, we choose ℓ such that $\mu \in \left(2^{-\ell-1}, 2^{-\ell}\right)$, and recall that we may assume that $\mu = a2^{-b}$ for some $a, b \in \mathbb{N}$ (since these points are dense within $[0,1]$). Choose $k \ge b$ and let F be a $(k, \ell+1)$-block parity. We consider the last block $x_1^{\ell+1}, \ldots, x_k^{\ell+1}$ of variables, and note that F contains 2^{k-1} clauses over this last block. Removing $0 \le t \le 2^{k-1}$ clauses over the last block of variables linearly changes μ from $2^{-\ell-1}$ at $t = 0$ to $2^{-\ell}$ at $t = 2^{k-1}$. Every

time we remove a clause from the last block, $2^{(k-1)\ell}$ formerly unsatisfying x become satisfying. Before removal, F at x was sensitive to the k variables in the last block (flipping them used to make x satisfying) whereas after removal, F at x is not sensitive to them anymore (change them and x will still satisfy F). However, F at x is now sensitive to the ℓk variables in the first ℓ blocks: changing any of them makes x unsatisfying. Thus, each time we remove a clasue, the number of edges from $\mathrm{sat}(F)$ to its complement in $\{0,1\}^n$ changes by the same amount. Therefore, $\mathbf{AS}(F)$ moves linearly from $2(\ell+1)k2^{-\ell-1}$ at $t=0$ to $2\ell k2^{-\ell}$ at $t=2^{k-1}$. At every step t, the point $(\mu(F), \mathbf{AS}(F)/k)$ lies on the line from $\left(2^{-\ell-1}, 2(\ell+1)2^{-\ell-1}\right)$ to $\left(2^{-\ell}, 2\ell 2^{-\ell}\right)$, *i.e.*, exactly on our upper bound curve. Choosing $t = a2^{k+\ell-b} - 2^{k+1}$ ensures F has density exactly $\mu = a2^{-b}$.

4 Average Sensitivity of Monotone k-CNFs

Revisiting Equation (2) in the proof of our upper bound in Section 3.1, recall that we used Jensen's inequality to handle the expression $\mathbf{E}_\pi[2^{I(x,\boldsymbol{\pi},F)}]$, where $I(x,\pi,F) = \sum_{i=1}^n I_i(x,\pi,F)$ is the number of inference steps made by the PPZ algorithm. The crux of our improvement for monotone k-CNFs is the observation that when F is monotone the indicator variables $I_1(x,\pi,F),\ldots,I_n(x,\pi,F)$ are *positively correlated*, *i.e.* $\mathbf{E}\left[2^I\right] \geq \prod_{i=1}^n \mathbf{E}\left[2^{I_i}\right]$, leading to a much better bound.

Lemma 4 (Positive correlation). *If F is monotone then the indicator variables $I_i(x,\pi,F)$ are positively correlated. That is, for every $x \in \mathrm{sat}(F)$,*

$$\mathop{\mathbf{E}}_{\pi}\left[2^{I(x,\boldsymbol{\pi},F)}\right] \geq \prod_{i=1}^{n} \mathop{\mathbf{E}}_{\pi}\left[2^{I_i(x,\boldsymbol{\pi},F)}\right]. \tag{4}$$

Proof of Theorem 2 assuming Lemma 4. We begin by analyzing each term in the product in the right-hand side of Equation (4). Let $x \in \mathrm{sat}(F)$ and $i \in [n]$. If F is sensitive to coordinate i at x then $\mathbf{Pr}_\pi[I_i(x,\boldsymbol{\pi},F)=1] \geq 1/k$ by Lemma 2, and so

$$\begin{aligned}
\mathop{\mathbf{E}}_{\pi}\left[2^{I_i(x,\boldsymbol{\pi},F)}\right] &= \mathop{\mathbf{Pr}}_{\pi}[I_i(x,\boldsymbol{\pi},F)=0]\cdot 1 + \mathop{\mathbf{Pr}}_{\pi}[I_i(x,\boldsymbol{\pi},F)=1]\cdot 2 \\
&= (1 - \mathbf{Pr}[I_i(x,\boldsymbol{\pi},F)=1]) + \mathbf{Pr}[I_i(x,\boldsymbol{\pi},F)=1]\cdot 2 \\
&= 1 + \mathbf{Pr}[I_i(x,\boldsymbol{\pi},F)=1] \;\geq\; 1 + \frac{1}{k}.
\end{aligned}$$

On the other hand if F is not sensitive to coordinate i at x, then $I_i(x,\pi,F)$ is always 0, and so $\mathbf{E}_\pi\left[2^{I_i(x,\boldsymbol{\pi},F)}\right] = 1$. Combining this with Lemma 4 shows that

$$\mathop{\mathbf{E}}_{\pi}\left[2^{I(x,\boldsymbol{\pi},F)}\right] \geq \prod_{i=1}^{n} \mathop{\mathbf{E}}_{\pi}\left[2^{I_i(x,\boldsymbol{\pi},F)}\right] \geq \left(1+\frac{1}{k}\right)^{S(F,x)}. \tag{5}$$

With this identity in hand Theorem 2 follows quite easily. Starting with Equation (2), we have

$$\begin{aligned}
1 &\geq \mu \mathop{\mathbf{E}}_{x\in\mathrm{sat}(F),\pi}\left[2^{I(\boldsymbol{x},\boldsymbol{\pi},F)}\right] && \text{(by Equation (2))}\\
&\geq \mu \mathop{\mathbf{E}}_{x\in\mathrm{sat}(F)}\left[\left(1+\frac{1}{k}\right)^{S(F,\boldsymbol{x})}\right] && \text{(by (5))}\\
&\geq \mu \left(1+\frac{1}{k}\right)^{\mathbf{E}_{x\in\mathrm{sat}(F)}[S(F,\boldsymbol{x})]} && \text{(by Jensen's inequality)}\\
&= \mu \left(1+\frac{1}{k}\right)^{\mathbf{AS}(F)/2\mu}. && \text{(by Fact 11)}
\end{aligned}$$

Solving for $\mathbf{AS}(F)$, we get

$$\mathbf{AS}(F) \leq \frac{2\mu\ln(1/\mu)}{\ln\left(1+\frac{1}{k}\right)} = \frac{2\mu\ln(1/\mu)\cdot k}{\ln\left(1+\frac{1}{k}\right)^k} = 2k\mu\ln(1/\mu)(1+\epsilon_k)\ ,$$

for some ϵ_k that goes to 0 as k grows. This proves Theorem 2. □

4.1 Proof of Lemma 4: Positive Correlation

Fix a satisfying assignment x of F. If F is insensitive to coordinate j on x (*i.e.* $S(F,x,j)=0$) then $I_j(x,\pi,F)=0$ for all permutations π, and so we first note that

$$\mathop{\mathbf{E}}_{\pi}\left[2^{I(x,\boldsymbol{\pi},F)}\right] = \mathop{\mathbf{E}}_{\pi}\left[\prod_{i:\ S(F,x,i)=1} 2^{I_i(x,\boldsymbol{\pi},F)}\right]. \tag{6}$$

Fix an i such that $S(F,x,i)=1$. At this point, it would be convenient to adopt the equivalent view of a random permutation π as a function $\pi:[n]\to[0,1]$ where we choose the value of each $\pi(k)$ independently and uniformly at random from $[0,1]$ (ordering $[n]$ according to π defines a uniformly random permutation). From this point of view $2^{I_i(x,\boldsymbol{\pi},F)}$ is a function from $[0,1]^n\to\{1,2\}$. The key observation that we make now is that the n functions $2^{I_i(x,\pi,F)}$ for $1\leq i\leq n$ are monotonically increasing in the coordinates at which x is 1, and decreasing in the coordinates at which x is 0.

By monotonicity, we know that $I_i(x,\pi,F)=1$ if only if there is a clause $C=x_i\vee x_{i_2}\vee\ldots\vee x_{i_{k'}}$ in F, where $x_{i_2}=\ldots=x_{i_{k'}}=0$ (note that $x_i=1$ by monotonicity) and $\pi(i_2),\pi(i_3),\ldots,\pi(i_{k'})<\pi(i)$. By this characterization, we see that

- Increasing $\pi(i)$ can only increase $I_i(x,\pi,F)$, and decreasing $\pi(i)$ can only decrease $I_i(x,\pi,F)$.
- Increasing $\pi(j)$ for some j where $x_j=0$ can only decrease $I_i(x,\pi,F)$, and decreasing $\pi(j)$ can only increase $I_i(x,\pi,F)$.
- Finally, $I_i(x,\pi,F)$ is not affected by changes to $\pi(j)$ when $j\neq i$ and $x_j=1$.

Therefore, the functions $2^{I_i(x,\pi,F)}$ where $S(F,x,i) = 1$ are all unate with the same orientation[5] and so by the FKG correlation inequality [27], they are positively correlated. We conclude that

$$\mathop{\mathbf{E}}_{\pi}\left[\prod_{i:\ S(F,x,i)=1} 2^{I_i(x,\boldsymbol{\pi},F)}\right] \geq \prod_{i:\ S(F,x,i)=1} \mathop{\mathbf{E}}_{\pi}\left[2^{I_i(x,\boldsymbol{\pi},F)}\right] = \prod_{i=1}^{n} \mathop{\mathbf{E}}_{\pi}\left[2^{I_i(x,\boldsymbol{\pi},F)}\right],$$

where in the final inequality we again use the fact that $I_j(x,\boldsymbol{\pi},F) = 0$ for all π if $S(F,x,j) = 0$. This proves Lemma 4.

4.2 Proof of Theorem 3: The Lower Bound

In this section we construct a monotone k-CNF formula with large average sensitivity. We will need a combinatorial identity.

Lemma 5. *Let $k, \ell \geq 0$. Then $\sum_{s=0}^{m} \binom{m}{s} k^s \ell^{m-s} s = mk(k+\ell)^{m-1}$.*

Proof. First, if $k + \ell = 1$, then both sides equal the expected number of heads in m coin tosses with head probability k. Otherwise, we divide both sides by $(k+\ell)^m$ and apply that argument to $\frac{k}{k+\ell}$ and $\frac{\ell}{k+\ell}$. □

Proof of Theorem 3. This function will be the tribes function $F := \mathrm{Tribes}_m^k$ over $n = km$ variables:

$$(x_1^{(1)} \vee x_2^{(1)} \cdots \vee x_k^{(1)}) \wedge (x_1^{(2)} \vee x_2^{(2)} \cdots \vee x_k^{(2)}) \wedge \cdots \wedge (x_1^{(m)} \vee x_2^{(m)} \cdots \vee x_k^{(m)})$$

This is a k-CNF formula with $\mathbf{E}[F] = \left(1 - 2^{-k}\right)^m$. We set $m := \left\lceil \ln(\mu) / \ln\left(1 - 2^{-k}\right) \right\rceil$, which yields $\mu\left(1 - 2^{-k}\right) \leq \mathbf{E}[F] \leq \mu$. Let us compute $\mathbf{AS}(F)$. For a satisfying assignment x, $S(F,x)$ is the number of clauses in Tribes_m^k containing exactly one satisfied literal. The number of satisfying assignments x of sensitivity s is exactly $\binom{m}{s} k^s \left(2^k - k - 1\right)^{m-s}$, as there are $2^k - k - 1$ ways to satisfy more than one literal in a k-clause. Thus,

$$\mathbf{AS}(F) = 2^{-n+1} \sum_{x \in \mathrm{sat}(F)} S(F,x) = 2^{-mk+1} \sum_{s=0}^{m} \binom{m}{s} k^s \left(2^k - k - 1\right)^{m-s} s$$

Applying Lemma 5 with $\ell = 2^k - k - 1$, we get

$$\mathbf{AS}(F) = 2^{-mk+1} mk (2^k - 1)^{m-1} = \frac{\left(2^k - 1\right)^m}{2^{km}} \frac{2mk}{2^k - 1} = \frac{2\,\mathbf{E}[F] mk}{2^k - 1}$$

Recall that $m \geq \frac{\ln(\mu)}{\ln(1-2^{-k})}$ and $\mathbf{E}[F] \geq \left(1 - 2^{-k}\right)\mu$. Thus,

$$\mathbf{AS}(F) = \frac{2\,\mathbf{E}[F] mk}{2^k - 1} \geq \frac{2k\mu\left(1 - 2^{-k}\right)\ln(\mu)}{(2^k - 1)\ln\left(1 - 2^{-k}\right)} = \frac{2k\mu \ln(\mu)}{2^k \ln(1 - 2^{-k})} = 2k\mu \ln\left(\frac{1}{\mu}\right)(1 - \varepsilon_k),$$

for some ε_k that quickly converges to 0 as k grows. This proves Theorem 3. □

[5] This means for each $1 \leq i \leq n$, they are either all monotonically increasing in $\pi(i)$ or all decreasing in $\pi(i)$.

4.3 A Gap between Average and Maximum Sensitivity

Recall that the maximum sensitivity $\mathbf{S}(F)$ of a Boolean function F is the quantity $\max_{x\in\{0,1\}^n}[S(F,x)]$. Clearly we have that $\mathbf{AS}(F) \leq \mathbf{S}(F)$, and this inequality is tight when $F = \mathsf{PAR}_n$, the parity function over n variables. Servedio conjectured that unlike the case for PAR_n, the average sensitivity of a *monotone* Boolean function F is always asymptotically smaller than its maximum sensitivity [28]:

Conjecture 1 (Servedio). There exists universal constants $K > 0$ and $\delta < 1$ such that the following holds. Let $F : \{0,1\}^n \to \{0,1\}$ be any monotone Boolean function. Then $\mathbf{AS}(F) \leq K \cdot \mathbf{S}(F)^\delta$.

In addition to being an interesting and natural question, Servedio's conjecture also has implications for Mansour's conjecture [31] on the Fourier spectrum of depth-2 AC^0, a longstanding open problem in analysis of Boolean functions and computational learning theory [28,32]. The conjecture can be checked to be true for the canonical examples of monotone Boolean functions such as majority ($\mathbf{AS}(\mathsf{MAJ}_n) = \Theta(\sqrt{n})$ whereas $\mathbf{S}(\mathsf{MAJ}_n) = \lceil n/2 \rceil$), and the Ben-Or-Linial Tribes function ($\mathbf{AS}(\mathsf{Tribes}_{k,2^k}) = \Theta(k)$ whereas $\mathbf{S}(\mathsf{Tribes}_{k,2^k}) = 2^k$). O'Donnell and Servedio have shown the existence of a monotone function F with $\mathbf{AS}(F) = \Omega(\mathbf{S}(F)^{0.61})$ [12], and this is the best known lower bound on the value of δ in Conjecture 1.

The current best separation between the two quantities is $\mathbf{AS}(F) \leq \sqrt{2/\pi} \cdot \mathbf{S}(F)(1 + o(1)) \approx 0.797 \cdot \mathbf{S}(F)(1 + o(1))$ where $o(1)$ is a term that tends to 0 as $\mathbf{S}(F)$ grows[6], which follows as a corollary of Blais's [29] sharpening of an isoperimetric inequality of O'Donnell and Servedio [12]. We now show that our upper bound in Theorem 2 yields an improved separation. We recall a basic fact from [33] characterizing the maximum sensitivity of a monotone Boolean function by its CNF and DNF widths:

Fact 41. *Let $F : \{0,1\}^n \to \{0,1\}$ be a monotone Boolean function. Then*

$$\mathbf{S}(F) = \max\{\mathbf{DNF\text{-}width}(F), \mathbf{CNF\text{-}width}(F)\}.$$

Corollary 1. *Let F be a monotone Boolean function. Then $\mathbf{AS}(F) \leq \ln(2) \cdot \mathbf{S}(F)(1 + o(1)) \leq 0.694 \cdot \mathbf{S}(f)(1 + o(1))$, where $o(1)$ is a term that goes to 0 as $\mathbf{S}(F)$ grows.*

Proof. By Fact 41, we have $\mathbf{CNF\text{-}width}(F) \leq \mathbf{S}(F)$ and $\mathbf{CNF\text{-}width}(\neg F) = \mathbf{DNF\text{-}width}(F) \leq \mathbf{S}(F)$. Applying the upper bound of Theorem 2 to both F and $\neg F$, we get

$$\mathbf{AS}(F) \leq \min\{2\mu \ln(1/\mu), 2(1-\mu)\ln(1/(1-\mu))\} \cdot \mathbf{S}(F)(1 + o(1)),$$

where $\mu = \mu(F)$. The proof is complete by noting that $\min\{2\mu \ln(1/\mu), 2(1-\mu)\ln(1/(1-\mu))\} \leq \ln(2)$ for all $\mu \in [0,1]$. □

[6] Note that this additive $o(1)$ term is necessary as $\mathbf{AS}(F) = \mathbf{S}(F) = 1$ for the monotone function $F(x) = x_1$.

5 Average Sensitivity of s-clause CNFs

Let F be computed by an s-clause CNF. It is straightforward to check that $\mathbf{Pr}[F(\boldsymbol{x}) \neq G(\boldsymbol{x})] \leq \varepsilon$ and $\mathbf{AS}(F) \leq \mathbf{AS}(G) + \varepsilon \cdot n$, if G is the CNF obtained from F by removing all clauses of width greater than $\log(s/\varepsilon)$. When $s = \Omega(n)$ we may apply Amano's theorem to G and take $\varepsilon = O(1/n)$ to conclude that $\mathbf{AS}(F) = O(\log s)$. Building on the work of Linial, Mansour and Nisan [7], Boppana employed Håstad's switching lemma to prove that in fact $\mathbf{AS}(F) = O(\log s)$ continues to hold for all values of $s = o(n)$. Here we give an elementary proof of Theorem 4 that sharpens and simplifies Boppana's result. A bound of $\mathbf{AS}(F) \leq \log(s)(1 + o(1))$, which is tight up to lower order terms by considering the parity function over $\log s$ variables, does not appear to have been known prior to this work.[7]

Proof of Theorem 4. We write $F = G \wedge H$, where G consists of all clauses of width at most τ and the threshold $\tau \in [s]$ will be chosen later. By the subadditivity of average sensitivity, we see that

$$\mathbf{AS}(F) \leq \mathbf{AS}(G) + \mathbf{AS}(H) \leq \tau + \sum_{C \in H} \mathbf{AS}(C) = \tau + \sum_{C \in H} \frac{|C|}{2^{|C|-1}} \leq \tau + s \cdot \frac{\tau}{2^{\tau-1}}.$$

Here the second inequality is by Amano's theorem applied to G and the subadditivity of average sensitivity applied to H, and the last inequality holds because $z \mapsto z/2^{z-1}$ is a decreasing function. Choosing $\tau := \log s + \log\log s$ yields $\mathbf{AS}(F) \leq \log s + \log\log s + 2 + o(1)$ and completes the proof. □

Acknowledgements. We thank Eric Blais and Homin Lee for sharing [29] and [30] with us. We also thank Rocco Servedio and Navid Talebanfard for helpful discussions.

References

1. Boppana, R.B.: The average sensitivity of bounded-depth circuits. Inf. Process. Lett. 63(5), 257–261 (1997) 1, 1, 1
2. Traxler, P.: Variable influences in conjunctive normal forms. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 101–113. Springer, Heidelberg (2009) 1, 1
3. Amano, K.: Tight bounds on the average sensitivity of k-CNF. Theory of Computing 7(4), 45–48 (2011) 1, 1
4. Paturi, R., Pudlák, P., Zane, F.: Satisfiability coding lemma. In: Proceedings of the 38th IEEE Symposium on Foundations of Computer Science, pp. 566–574 (1997) 1, 2, 1, 2
5. Ben-Or, M., Linial, N.: Collective coin flipping. In: Micali, S., Preparata, F. (eds.) Randomness and Computation. Advances in Computing Research: A research annual, vol. 5, pp. 91–115. JAI Press (1990) 1

[7] Working through the calculations in Boppana's proof one gets a bound of $\mathbf{AS}(F) \leq 3 \log s$.

6. Kahn, J., Kalai, G., Linial, N.: The influence of variables on Boolean functions. In: Proceedings of the 29th Annual Symposium on Foundations of Computer Science, pp. 68–80 (1988) 1
7. Linial, N., Mansour, Y., Nisan, N.: Constant depth circuits, Fourier transform and learnability. Journal of the ACM 40(3), 607–620 (1993) 1, 5
8. O'Donnell, R., Wimmer, K.: Approximation by DNF: Examples and counterexamples. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 195–206. Springer, Heidelberg (2007) 1
9. Blais, E., Tan, L.Y.: Approximating Boolean functions with depth-2 circuits. In: Conference on Computational Complexity (2013) 1
10. Khrapchenko, V.M.: A method of determining lower bounds for the complexity of π-schemes. Math. Notes Acad. Sci. USSR 10(1), 474–479 (1971) 1
11. Bshouty, N., Tamon, C.: On the Fourier spectrum of monotone functions. Journal of the ACM 43(4), 747–770 (1996) 1
12. O'Donnell, R., Servedio, R.: Learning monotone decision trees in polynomial time. SIAM Journal on Computing 37(3), 827–844 (2008) 1, 4.3
13. Diakonikolas, I., Harsha, P., Klivans, A., Meka, R., Raghavendra, P., Servedio, R., Tan, L.Y.: Bounding the average sensitivity and noise sensitivity of polynomial threshold functions. In: Proceedings of the 42nd Annual ACM Symposium on Theory of Computing, pp. 533–542 (2010) 1
14. Friedgut, E.: Boolean functions with low average sensitivity depend on few coordinates. Combinatorica 18(1), 27–36 (1998) 1
15. Friedgut, E.: Sharp thresholds of graph properties, and the k-SAT problem. Journal of the American Mathematical Society 12(4), 1017–1054 (1999) 1
16. Benjamini, I., Kalai, G., Schramm, O.: Noise sensitivity of Boolean functions and applications to percolation. Publications Mathématiques de l'IHÉS 90(1), 5–43 (1999) 1
17. Dinur, I., Safra, S.: On the hardness of approximating minimum vertex cover. Annals of Mathematics 162(1), 439–485 (2005) 1
18. Shi, Y.: Lower bounds of quantum black-box complexity and degree of approximating polynomials by influence of boolean variables. Inform. Process. Lett. 75(1-2), 79–83 (2000) 1
19. Ron, D., Rubinfeld, R., Safra, M., Samorodnitsky, A., Weinstein, O.: Approximating the influence of monotone boolean functions in $o(n)$ query complexity. TOCT 4(4), 11 (2012) 1
20. Harper, L.: Optimal assignments of numbers to vertices. Journal of the Society for Industrial and Applied Mathematics 12(1), 131–135 (1964) 1
21. Bernstein, A.: Maximally connected arrays on the n-cube. SIAM Journal on Applied Mathematics, 1485–1489 (1967) 1
22. Lindsey, J.H.: Assignment of numbers to vertices. Amer. Math. Monthly 71, 508–516 (1964) 1
23. Hart, S.: A note on the edges of the n-cube. Discrete Mathamatics 14(2), 157–163 (1976) 1
24. Mansour, Y.: An $O(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution. Journal of Computer and System Sciences 50(3), 543–550 (1995) 1
25. Håstad, J.: A slight sharpening of LMN. Journal of Computer and System Sciences 63(3), 498–508 (2001) 1
26. O'Donnell, R.: Lecture 29: Open problems. Scribe notes for a course on Analysis of Boolean Functions at Carnegie Mellon University (2007) 1
27. Fortuin, C.M., Kasteleyn, P.W., Ginibre, J.: Correlation inequalities on some partially ordered sets. Comm. Math. Phys. 22, 89–103 (1971) 1.1, 4.1

28. O'Donnell, R.: Open problems in analysis of boolean functions. CoRR abs/1204.6447 (2012) 1.1, 4.3, 4.3
29. Blais, E.: Personal communication (2011) 1.1, 4.3, 5
30. Lee, H.: Personal communication (2012) 4, 5
31. Mansour, Y.: Learning Boolean functions via the Fourier Transform. In: Roychowdhury, V., Siu, K.Y., Orlitsky, A. (eds.) Theoretical Advances in Neural Computation and Learning, pp. 391–424. Kluwer Academic Publishers (1994) 4.3
32. Gopalan, P., Kalai, A., Klivans, A.: A query algorithm for agnostically learning DNF? In: Proceedings of the 21st Annual Conference on Learning Theory, pp. 515–516 (2008) 4.3
33. Nisan, N.: CREW PRAMs and decision trees. SIAM Journal on Computing 20(6), 999–1007 (1991) 4.3

Improved Bounds on the Phase Transition for the Hard-Core Model in 2-Dimensions

Juan C. Vera[1], Eric Vigoda[2,⋆], and Linji Yang[2,⋆]

[1] Department of Econometrics and Operations Research,
Tilburg University, 5000 LE Tilburg, The Netherlands
j.c.veralizcano@uvt.nl
[2] School of Computer Science, Georgia Institute of Technology,
Atlanta GA 30332, USA
{vigoda,ljyang}@cc.gatech.edu

Abstract. For the hard-core lattice gas model defined on independent sets weighted by an activity λ, we study the critical activity $\lambda_c(\mathbb{Z}^2)$ for the uniqueness threshold on the 2-dimensional integer lattice $\mathbb{Z}^2$. The conjectured value of the critical activity is approximately 3.796. Until recently, the best lower bound followed from algorithmic results of Weitz (2006). Weitz presented an FPTAS for approximating the partition function for graphs of constant maximum degree Δ when $\lambda < \lambda_c(\mathbb{T}_\Delta)$ where $\mathbb{T}_\Delta$ is the infinite, regular tree of degree Δ. His result established a certain decay of correlations property called strong spatial mixing (SSM) on $\mathbb{Z}^2$ by proving that SSM holds on its self-avoiding walk tree $T_{\text{saw}}(\mathbb{Z}^2)$, and as a consequence he obtained that $\lambda_c(\mathbb{Z}^2) \geq \lambda_c(\mathbb{T}_4) = 1.675$. Restrepo et al. (2011) improved Weitz's approach for the particular case of $\mathbb{Z}^2$ and obtained that $\lambda_c(\mathbb{Z}^2) > 2.388$. In this paper, we establish an upper bound for this approach, by showing that SSM does not hold on $T_{\text{saw}}(\mathbb{Z}^2)$ when $\lambda > 3.4$. We also present a refinement of the approach of Restrepo et al. which improves the lower bound to $\lambda_c(\mathbb{Z}^2) > 2.48$.

Keywords: Hard-core Model, Uniqueness, Phase Transition, Strong Spatial Mixing, Approximate Counting.

1 Introduction

The hard-core model is a model of a gas composed of particles of non-negligible size and consequently configurations of the model are independent sets [4,8]. For a (finite) graph $G = (V, E)$ and an activity $\lambda > 0$ (corresponding to the fugacity of the gas), configurations of the model are the set Ω of independent sets of G where $\sigma \in \Omega$ has weight $w(\sigma) = \lambda^{|\sigma|}$. The Gibbs measure is defined as $\mu(\sigma) = w(\sigma)/Z$ where $Z = \sum_{\eta \in \Omega} w(\eta)$ is the partition function.

A fundamental question for statistical physics models, such as the hard-core model, is whether there exists a unique or there are multiple infinite-volume Gibbs measures on $\mathbb{Z}^2$. An equivalent question is whether the influence of the

⋆ Research supported in part by NSF grant CCF-1217458.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 699–713, 2013.

boundary on the origin decays in the limit. More formally, for a box in $\mathbb{Z}^2$ of side length $2L+1$ centered around the origin, let p_L^{even} (p_L^{odd}) denote the marginal probability that the origin is unoccupied conditional on the even (odd, respectively) vertices on the boundary being occupied. If

$$\lim_{L\to\infty} \left|p_L^{\text{odd}} - p_L^{\text{even}}\right| = 0$$

then there is a unique Gibbs measure on $\mathbb{Z}^2$, and if this limit is > 0 then there are multiple Gibbs measures. It is believed that there is a critical activity $\lambda_c(\mathbb{Z}^2)$ such that for $\lambda < \lambda_c(\mathbb{Z}^2)$ uniqueness holds, and for $\lambda > \lambda_c(\mathbb{Z}^2)$ non-uniqueness holds. For the infinite, regular tree $\mathbb{T}_\Delta$ of degree Δ it is easy to show that $\lambda_c(\mathbb{T}_\Delta) = (\Delta-1)^{\Delta-1}/(\Delta-2)^\Delta$ [9].

There are long-standing heuristic results which suggest that $\lambda_c(\mathbb{Z}^2) \approx 3.796$ [8,2,10]. For the upper bound on the critical activity, a classical Peierls' type argument implies $\lambda_c(\mathbb{Z}^2) = O(1)$ [7], and Blanca et al. [5] improved this upper bound to show $\lambda_c(\mathbb{Z}^2) < 5.3646$. Our focus is on the lower bound.

Weitz [15] showed that $\lambda_c(\mathbb{Z}^2) \geq \lambda_c(\mathbb{T}_4) = 27/16 = 1.6875$. His result followed from the algorithmic result. For all graphs with constant maximum degree Δ, $\lambda < \lambda_c(\mathbb{T}_\Delta)$, Weitz [15] presented an FPTAS for approximating the partition function. A central step in his approach is proving a certain decay of correlations property known as *strong spatial mixing (SSM)* on the graph G. SSM says that for every $v \in V$, every $T \subset V$ and $S \subset T$, and pair of configurations σ, τ on T which only differ on S (i.e., $\sigma(T \setminus S) = \tau(T \setminus S)$) then the difference in the influence of σ and τ on the marginal probability of v decays exponentially in the distance of v from the difference set S (see Section 2 for formal definitions of these concepts). In contrast, *weak spatial mixing (WSM)* only requires that the influence decays exponentially in the distance to the set T. For the hard-core model, since fixing a vertex to be unoccupied or occupied can be realized by removing the vertex or the vertex and its neighbors, it then follows that SSM on a graph G is equivalent to WSM for all (vertex induced) subgraphs of G.

Weitz constructs a version of the tree $T_{\text{saw}}(G, v)$ of self-avoiding walks from $v \in V$, in such a way that SSM on $T_{\text{saw}}(G, v)$ for all v implies SSM on G. His variant of the self-avoiding walk tree fixes the leaves of the tree (corresponding to the walk completing a cycle in G) to be occupied or unoccupied based on a fixed, arbitrary ordering of the neighbors for each vertex. He then shows that SSM holds on the complete tree $\mathbb{T}_\Delta$, and hence SSM holds on all trees of maximum degree Δ when $\lambda < \lambda_c(\mathbb{T}_\Delta)$.

Restrepo et al. [11] improve upon Weitz's approach for $\mathbb{Z}^2$ by utilizing its structure to build a better "bounding tree" than $\mathbb{T}_\Delta$. They define a set of branching matrices $\mathbf{M}_\ell$ for $\ell \geq 4$ corresponding to walks in $\mathbb{Z}^2$ containing no cycles of length $\leq \ell$ (see Section 3 for a more formal introduction to these notions). The key point is that $T_{\text{saw}}(\mathbb{Z}^2)$ is a subtree of the tree $T_{\mathbf{M}_\ell}$ defined by $\mathbf{M}_\ell$. They then present a decay of correlation proof by using a suitable message passing approach for proving SSM for $T_{\mathbf{M}_\ell}$, and hence for $T_{\text{saw}}(\mathbb{Z}^2)$ as well. They show that SSM holds on $T_{\mathbf{M}_6}$ for $\lambda < 2.33$, and SSM holds on $T_{\mathbf{M}_8}$ for $\lambda < 2.388$. Consequently, they establish that $\lambda_c(\mathbb{Z}^2) > 2.388$.

Our first result establishes a limit to these approaches by showing that SSM does not hold on $T_{\text{saw}}(\mathbb{Z}^2)$. As mentioned earlier, in the construction of $T_{\text{saw}}(\mathbb{Z}^2)$, the assignment for leaves depends on the ordering in $\mathbb{Z}^2$ of neighbors of each vertex. Since $\mathbb{Z}^2$ is vertex-transitive, it is natural to define an ordering that is identical for every vertex (e.g., based on an ordering of the directions N, S, E, and W), which we refer to as a *homogenous ordering*. We prove the following result.

Theorem 1. *For $T_{\text{saw}}(\mathbb{Z}^2)$, SSM does not hold when $\lambda > 3.4$. Moreover, when $T_{\text{saw}}(\mathbb{Z}^2)$ is based on a homogenous ordering, then SSM does not hold when $\lambda > 3$.*

The theorem follows from considering a tree T that is a subtree of $T_{\text{saw}}(\mathbb{Z}^2)$ and establishing the threshold for WSM on T. The tree T that we consider in the homogenous ordering case is quite simple. When N is first in the ordering, the tree is simply the never-go-south tree (see Section 4). For any $T_{\text{saw}}(\mathbb{Z}^2)$ that is based on an inhomogeneous ordering, we are able to find another general subtree for which the WSM does not hold when $\lambda = 3.4$. Such an example gives a strong evidence that in order to prove the SSM for $\mathbb{Z}^2$ when λ is close to the conjectured threshold, the self-avoiding walk tree approach might not be appropriate. There are subtrees of the SAW tree of $\mathbb{Z}^2$ that have lower WSM threshold and hence one has to figure out an approach to exclude such trees.

We then present an improvement of the approach of Restrepo et al. [11] for proving SSM for the trees $T_{\mathbf{M}_\ell}$. They consider a particular statistic of the marginal distributions of the vertices, and prove the correlation decay property inductively on the height. The statistics can be viewed as a message passing algorithm, a variant of belief propagation. The messages they consider are a natural generalization of the message which is used to analyze the complete tree up to the tree threshold $\lambda_c(\mathbb{T}_\Delta)$ (which thereby reproves Weitz's result [15]). They establish a so-called DMS condition as a sufficient condition for these messages to imply SSM holds on the tree under consideration. Some of the limitations of their approach are that to find the settings for the parameters in their messages and the DMS condition, they use a heuristic hill-climbing algorithm which might become trapped in local optima. In addition, verifying their DMS condition is non-trivial.

In this paper, we consider piecewise linear functions for the messages. As a consequence, we can find these functions by solving a linear program. This yields improved results and simpler proofs of the desired contraction property. Consequently, we prove SSM holds for $T_{\mathbf{M}_6}$ when $\lambda \leq 2.45$ (previously, 2.33 by the DMS condition) and SSM holds for $T_{\mathbf{M}_8}$ when $\lambda \leq 2.48$ (previously, 2.388). This establishes the following theorem.

Theorem 2. $\lambda_c(\mathbb{Z}^2) > 2.48$.

The rest of the paper is organized in the following way. We formally define WSM and SSM in Section 2 and also present there the self-avoiding walk tree construction used by Weitz [15]. In Section 3, we will introduce branching matrices and present the framework of Restrepo et al. [11] in a manner tailored to our work. In Section 4 we will discuss limitations of Weitz's approach by showing several

counter-examples. Finally, in Section 5 we discuss our linear programming approach for proving SSM, which yields an improvement on the lower bound for the uniqueness threshold of the hard-core model on $\mathbb{Z}^2$.

2 Preliminaries

2.1 Definitions of WSM and SSM

For a graph $G = (V, E)$ and $S \subset V$, we define the boundary condition σ on S to be a fixed configuration on S. For a boundary condition σ, let $p_v(\sigma)$ be the unoccupied probability of vertex v in the Gibbs distribution μ on G conditional on σ. We now formally define WSM and SSM.

Definition 1 (Weak Spatial Mixing). *For the hard-core model at activity λ, for finite graph $G = (V, E)$,* WSM *holds if there exists $0 < \gamma < 1$ such that for every $v \in V$, every $S \subset V$, and every two configurations σ_1, σ_2 on S,*

$$|p_v(\sigma_1) - p_v(\sigma_2)| \ \leq \ \gamma^{\mathrm{dist}(v,S)}$$

where $\mathrm{dist}(v, S)$ *is the graph distance (i.e., length of the shortest path) between v and (the nearest point in) the subset S.*

For an infinite graph G, we define the WSM threshold for G as

$$WSM(G) = \inf\{\lambda : \text{WSM does not hold on } G \text{ at activity } \lambda\}.$$

Definition 2 (Strong Spatial Mixing). *For the hard-core model at activity λ, for finite graph $G = (V, E)$,* SSM *holds if there exists a $0 < \gamma < 1$ such that for every $v \in V$, every $S \subset V$, every $S' \subset S$, and every two configurations σ_1, σ_2 on S where $\sigma_1(S \setminus S') = \sigma_2(S \setminus S')$,*

$$|p_v(\sigma_1) - p_v(\sigma_2)| \ \leq \ \gamma^{\mathrm{dist}(v,S')}.$$

Finally, let $SSM(G)$ denote the SSM threshold for G, defined analogously to $WSM(G)$ but with respect to SSM.

To contrast the definitions of WSM and SSM, note that in WSM the influence decays exponentially in the distance to the boundary set S, whereas in SSM it is exponentially in the distance to the subset S' of the boundary that they differ on. An important observation that we repeat from the Introduction to emphasize it, is that for the hard-core model, for a tree T, SSM holds if and only if for all subtrees of T WSM holds.

2.2 Weitz's SAW Tree

We now detail Weitz's self-avoiding walk tree construction [15]. Given $G = (V, E)$, we fix an arbitrary ordering $>_w$ on the neighbors of each vertex w in G. For each $v \in V$, the tree $T_{\mathrm{saw}}(G, v)$ rooted at v is constructed as follows.

Consider the tree T of self-avoiding walks originating from v, including the vertices closing a cycle in the walks as leaves. We assign a boundary condition to the leaves by the following rule. Each leaf closes a cycle in G, so say the leaf corresponds to vertex w in G and the path leading to the leaf corresponds to the path $w \to v_1 \to \ldots v_\ell \to w$ in G. Then if $v_1 >_w v_\ell$ we fix this leaf to be unoccupied, and if $v_1 <_w v_\ell$ we fix this leaf to be occupied. Since we are in the hard core model, if the leaf is fixed to be unoccupied we simply remove that vertex from the tree. And if the leaf is fixed to be occupied, we remove that leaf and all of its neighbors from the tree, i.e., we remove completely the subtree rooted at the parent of that leaf.

If a boundary condition Γ is assigned to a subset S of G, then the self-avoiding walk tree can also be constructed consistently to the boundary condition, i.e., for a vertex $w \in S$ of G, we assign $\Gamma(w)$ to every occurrence of w in $T_{\text{saw}}(G, v)$. Weitz proves that, for any boundary condition on G and any vertex v, the marginal distribution of v on G is the same as the marginal distribution of the root of $T_{\text{saw}}(G, v)$ with the corresponding boundary condition. This further implies the following.

Lemma 1 (Weitz [15]). *For a specific λ, if for all v, SSM holds for $T_{\text{saw}}(G, v)$, then SSM holds for G.*

3 Message Passing Approach for Proving SSM

Let us first recall the recurrence of the marginal distributions on trees for the hard-core model. For now, we fix our infinite tree to be T. Let v be a vertex of T, and let T_v denote the subtree of T rooted at v. Let $N^-(v)$ denote the children of v in T_v. Let $\alpha_v(\Gamma)$ be the unoccupied probability of vertex v in the subtree T_v rooted at v with boundary condition Γ. It is straightforward to establish that $\alpha_v(\Gamma)$ satisfy the following recurrence:

$$\alpha_v(\Gamma) = \frac{1}{1 + \lambda \prod_{w \in N^-(v)} \alpha_w(\Gamma)}. \tag{1}$$

There are two special boundary conditions: one is called the odd boundary condition (denoted as $\Gamma_{o,L}$) which occupies all the vertices at level L when L is odd (and unoccupies when L is even); the other is called the even boundary condition (denoted as $\Gamma_{e,L}$) which occupies all the vertices at level L when L is even (and unoccupies when L is odd). These two boundary conditions are the extremal ones, meaning that for any other boundary condition Γ for the vertices at distance L from the root r of T, $\alpha_r(\Gamma_{e,L}) \leq \alpha_r(\Gamma) \leq \alpha_r(\Gamma_{o,L})$ when L is even (and with the inequalities reversed when L is odd).

To see that WSM holds for the tree T, it is enough to show that for the odd and even boundary conditions $\{\Gamma_{o,L}\}_{L \in \mathbb{N}}$ and $\{\Gamma_{e,L}\}_{L \in \mathbb{N}}$, the difference of the marginal probabilities at the root $|\alpha_r(\Gamma_{o,L}) - \alpha_r(\Gamma_{e,L})|$ decay exponentially in L.

3.1 Branching Matrices

Recall that in order to show that uniqueness holds for $\mathbb{Z}^2$ for a certain λ, it is enough to show that for the same λ, SSM holds on a certain tree which is a super-tree of $T_{\text{saw}}(\mathbb{Z}^2)$. Due to the regularity of $\mathbb{Z}^2$, in [11], deterministic multi-type Galton-Watson trees are proposed to characterize the candidate super-trees. The trees can be defined by matrices in the following way.

Definition 3. *Given a $t \times t$ (branching) matrix $\mathbf{M}$, $\mathcal{F}_{\leq \mathbf{M}}$ is the family of trees which can be generated under the following restrictions:*

- *Each vertex in tree $T \in \mathcal{F}_{\leq \mathbf{M}}$ has its type $i \in \{1, \ldots, t\}$.*
- *Each vertex of type i has at most M_{ij} children of type j.*

We use $T_{\mathbf{M}}$ to refer to the tree that is generated by the matrix $\mathbf{M}$, specifically, we mean the largest tree in the family $\mathcal{F}_{\leq \mathbf{M}}$. The simplest $\mathbf{M}$ such that $T_{\text{saw}}(\mathbb{Z}^2)$ is in the family $\mathcal{F}_{\mathbf{M}}$ is $\mathbf{M} = \begin{pmatrix} 0 & 4 \\ 0 & 3 \end{pmatrix}$. In this case, $\mathbb{T}_{\mathbf{M}}$ is the complete, regular tree of degree 4. As shown in [11], because of the regularity of $\mathbb{Z}^2$, a more sophisticated set of branching matrices $\mathbf{M}'$ we contain $T_{\text{saw}}(\mathbb{Z}^2)$ in their family are trees $T_{\mathbf{M}'}$ corresponding to all walks of $\mathbb{Z}^2$ truncated when closing a cycle of length less than or equal to a certain constant. Clearly, $T_{\mathbf{M}'}$ is a super-tree of $T_{\text{saw}}(\mathbb{Z}^2)$, because any path in $T_{\mathbf{M}'}$ will only avoid cycles of a certain length whereas paths in $T_{\text{saw}}(\mathbb{Z}^2)$ are avoiding all cycles.

When one tries to avoid a cycle of length 4, the matrix becomes

$$\mathbf{M}'_4 = \begin{pmatrix} 0 & 4 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix},$$

where each type is simply representing the various stages of completing a cycle of length 4 in a walk. It is easy to verify that $T_{\text{saw}}(\mathbb{Z}^2)$ is in the family $\mathcal{F}_{\mathbf{M}'_4}$.

In $\mathbf{M}'_4$, we have not yet taken into consideration the effect of the assignments to leaves as detailed in the construction of T_{saw} in Section 2.2. When we do that, we are able to construct much more sophisticated branching matrices which yield better bounds. Therefore, for $\ell \geq 4$, let $\mathbf{M}_\ell$ denote the branching matrix generating the tree containing all walks in $\mathbb{Z}^2$ truncated when completing a cycle of length $\leq \ell$, where these leaf vertices are occupied or unoccupied according to the definition in Section 2.2 based on some fixed homogeneous ordering $<_w$ of neighbors for every vertex. By taking into account the boundary condition we obtain a smaller tree since when a walk closes a cycle with an occupied assignment to a vertex u, this forces the parent of u to be unoccupied, which further trims down the size of the tree. These more sophisticated matrices yield a "tighter" bound on $T_{\text{saw}}(\mathbb{Z}^2)$, however the number of types increase. For example, for $\ell = 4$, whereas $\mathbf{M}'_4$ has 4 types, $\mathbf{M}_4$ has 17 types (after some simplifications), see [11] for details of $\mathbf{M}_4$. For $\mathbf{M}_6$ there are 132 types, and for $\mathbf{M}_8$ there are 922 types.

3.2 Contraction Principle

For each t by t branching matrix $\mathbf{M}$, we would like to derive a condition such that SSM holds for the tree $T_{\mathbf{M}}$. Throughout this paper, for each type i, we treat the row $\mathbf{M}_i$ of $\mathbf{M}$ as a multi-set and each entry $\mathbf{M}_i(j)$ of the row denotes the number of elements the set $\mathbf{M}_i$ has of type j. We use $t(w)$ to denote the type of vertex $w \in \mathbf{M}_i$. The following lemma, which is re-stating Lemma 1 from [11] in a slightly simpler form that is more convenient for our work, provides a sufficient condition for SSM to hold for the tree $T_{\mathbf{M}}$.

Lemma 2. *Let a branching matrix* $\mathbf{M}$ *be given. Assume there is* $0 < \gamma < 1$ *such that for each type* i*, there is a positive integrable function* Ψ_i *where*

$$\frac{1-\alpha_i}{\Psi_i(\alpha_i)} \sum_{w \in \mathbf{M}_i} \Psi_{t(w)}(\alpha_w) < \gamma, \tag{2}$$

for α_w *in the range* $[1/(1+\lambda), 1]$ *for each child* w *and* $\alpha_i = \left(1 + \lambda \prod_{w \in \mathbf{M}_i} \alpha_w\right)^{-1}$ *defined in* (1) *as a function of* α_w*'s. Then SSM holds for* $T_{\mathbf{M}}$*, i.e., WSM holds for all trees* T *in the family* $\mathcal{F}_{\leq \mathbf{M}}$ *with a fixed rate* $\gamma < 1$*.*

The proof of the lemma is included in the full version of this paper [13].

4 Upper Bound on the SSM Threshold

As described in the introduction, previous approaches for lower bounding $\lambda_c(\mathbb{Z}^2)$ are based on proving SSM for $T_{\text{saw}}(\mathbb{Z}^2)$. To provide a bound on the strength of these approaches we upper bound the SSM threshold for $T_{\text{saw}}(\mathbb{Z}^2)$. We will show that for $\lambda \geq 3.4$, SSM does not hold for $T_{\text{saw}}(\mathbb{Z}^2)$ obtained from any edge-ordering used in the vertices. We also show the for any homogeneous ordering SSM does not hold for $T_{\text{saw}}(\mathbb{Z}^2)$ for $\lambda \geq 3$. Note that this does not imply anything about WSM/SSM on $\mathbb{Z}^2$, it simply shows a limitation on the power of the current proof approaches.

To prove that SSM does not hold on $T_{\text{saw}}(\mathbb{Z}^2)$ we define a tree T that is a subtree of $T_{\text{saw}}(\mathbb{Z}^2)$ and prove that WSM does not hold on T for sufficiently large λ.

4.1 Upper Bound for Homogenous Ordering

We define a branching matrix D_H such that T_{D_H} corresponds to the never-go-South tree, and prove that WSM does not hold on this tree when $\lambda > 3$.

Since we are assuming a homogeneous ordering, without loss of generality assume that N is smallest in the ordering. We construct D_H by considering those walks on $\mathbb{Z}^2$ that only go N, E, and W. The branching rules can be written in the following finite state machine way:

$$0.\ O \to N \mid E \mid W, \quad 1.\ N \to N \mid E \mid W, \quad 2.\ E \to N \mid E, \quad 3.\ W \to N \mid W,$$

where O corresponds to the origin and is a transient state so can be ignored when analyzing the recurrence. The branching matrix corresponding to the above rule is

$$D_H = \begin{pmatrix} 1\,1\,1 \\ 1\,1\,0 \\ 1\,0\,1 \end{pmatrix}, \tag{3}$$

where rows/columns 1, 2, and 3 correspond to North, East, and West respectively.

Lemma 3. *Let the order of the edges in each vertex be an homogenous order where N is the smallest in the order. The tree T_{D_H} generated by the branching matrix D_H is a subtree of $T_{\text{saw}}(\mathbb{Z}^2)$.*

For the tree T_{D_H} we can establish its WSM threshold as stated in the following result, which immediately implies Theorem 1.

Lemma 4

$$WSM(T_{D_H}) = 3.$$

The proofs of Lemmas 3 and 4 are included in the full version of this paper [13].

4.2 Ordering-independent Subtree for T_{saw}

In this section, we will define a branching matrix D_G such that the generated tree T_{D_G} is a subtree of T_{saw} independently on the ordering of edges for each vertex. When in particular the ordering is homogeneous, then T_{D_G} it is a subtree of the tree T_{D_H} defined in the previous section. This new tree T_{D_G} also never goes South (as in T_{D_H}) but it has further structure to ensure that the leaves in this tree are at least distance two from the leaves of the self-avoiding walk tree $T_{\text{saw}}(\mathbb{Z}^2)$, and therefore for any boundary condition for $T_{\text{saw}}(\mathbb{Z}^2)$ (and hence any ordering of the edges for each vertex) it immediately follows that T_{D_G} is a subtree of $T_{\text{saw}}(\mathbb{Z}^2)$. To achieve this property, we add to the never-go-South tree the rule that if the walk goes East there must be at least two North steps before it goes West (and similarly, for West to East).

The tree is constructed by the following rules:

0. $O \to N \mid E \mid W$,

1. $N \to NN \mid NE \mid NW$, 2. $W \to WN \mid WW$, 3. $E \to EN \mid EE$,
4. $NN \to NN \mid NE \mid NW$, 5. $NW \to WN \mid WW$, 6. $NE \to EN \mid EE$,
7. $WW \to WN \mid WW$, 8. $EE \to EN \mid EE$, 9. $WN \to NW \mid NN$,
10. $EN \to NE \mid NN$.

Here the state O corresponds to the origin, while E, W and N correspond to the first edges in the path. Then each of the states corresponds to the last two visited edges. Notice also that states O, E, W and N are transient states. We denote the branching matrix for this tree as D_G.

Lemma 5. *The tree T_{D_G} generated by the branching matrix D_G is a subtree of $T_{\text{saw}}(\mathbb{Z}^2)$, independently of the edge ordering used for each vertex.*

We establish the following bounds on the WSM threshold for the tree T_{D_G}.

Lemma 6

$$3.3 < WSM(T_{D_G}) < 3.4.$$

The proofs of Lemmas 5 and 6 are included in the full version of this paper [13]. Theorem 1 follows from Lemmas 4 and 6.

4.3 Tree with Different Thresholds for SSM and WSM

Brightwell et al. [6] give an example of a tree for which WSM holds but SSM does not hold for the same activity λ. Here, we present another example which is more closely related to $T_{\text{saw}}(\mathbb{Z}^2)$. We show a tree T', which is a super-tree of T_{D_H} and subtree of $T_{\text{saw}}(\mathbb{Z}^2)$, for which WSM holds for some $\lambda > 3$.

To construct the tree T' we allow some South moves in the tree in a certain context. In particular, we only allow that a South move happens when the path contains the following substring: NNEESEN, i.e., a South move is allowed if and only if it is after a sequence of NNEE moves and followed by EN moves.

We will prove that the WSM threshold for $T' = T_{D'}$, the tree generated by D' is above $\lambda = 3.01$, and hence, combined with Lemma 4, we get the following lemma. The tree family can be formalized in the following finite state machine way:

1. $E \to E \mid N$
2. $W \to N \mid W$
3. $N \to NN \mid E \mid W$
4. $NN \to NN \mid NNE \mid W$
5. $NNE \to N \mid NNEE$
6. $NNEE \to N \mid E \mid NNEES$
7. $NNEES \to NNEESE$
8. $NNEESE \to NNEESEN$
9. $NNEESEN \to NN \mid E$

Let the matrix describing the above rules be denoted as D'.

Lemma 7. *For the tree $T_{D'}$ at $\lambda = 3.01$, WSM holds but SSM does not hold.*

The proof of this lemma can be found in the full version of this paper [13].

5 Linear Program for Lower Bounding SSM Threshold

Here we propose a way to use linear programming to solve the functional inequality (2). Notice that if Ψ_i is positive and bounded for all i then inequality (2) is equivalent to

$$(1 - \alpha_i) \sum_{w \in \mathbf{M}_i} \Psi_{t(w)}(\alpha_w) < \Psi_i(\alpha_i). \tag{4}$$

The idea to solve (4) is simple. We will restrict the search for Ψ_i to a family of positive piecewise linear functions with a finite number of discontinuities.

First of all, it is a simple fact that each α_i is in the interval $I = [1/(1+\lambda), 1]$. We will divide I into a set of d consecutive sub-intervals of the same size. Define

$$X_k = \frac{1}{1+\lambda} + k\frac{\lambda}{d(1+\lambda)}, \text{ for } k = 0, \ldots, d-1.$$

To ease the notation define $Y_k = X_{k+1}$ for $k = 0, \ldots, d-1$. Note that the intervals $[X_k, Y_k]$ partition I. Since the only requirements of $\Psi_i(x)$ are positive and integrable, we restrict the search for $\Psi_i(x)$ to functions of linear form $-a_{i,k}x + b_{i,k}$ in each interval $[X_k, Y_k]$ with $a_{i,k}, b_{i,k} > 0$.

Now, for each type i, the functional inequality can be decomposed according to different combinations of the intervals of the variables α_w which are type i's children. For each combination, we are able to write down a set of linear inequalities such that it is a sufficient condition for the functional inequality to hold within that region.

To capture for which sub-intervals should (4) hold, we say that a tuple of indexes $k = (k_0, k_1, k_2, \ldots, k_{\Delta_i})$ is i-acceptable if the interval $[X_{k_0}, Y_{k_0}]$ intersects the interval $\left[\frac{1}{1+\lambda\prod_{j=1}^{\Delta_i} Y_{k_j}}, \frac{1}{1+\lambda\prod_{j=1}^{\Delta_i} X_{k_j}}\right]$. We have the following theorem.

Theorem 3. *In order for the functional inequality* (2) *to hold, it is enough for the following set of linear constraints (a's and b's are the variables) to be feasible:*

For each $i \in [t]$ and each i-acceptable tuple k,

$$(1 - X_{k_0}) \sum_{j=1}^{\Delta_i} \left(b_{t(j),k_j} - a_{t(j),k_j} X_{k_j}\right) < \left(b_{i,k_0} - a_{i,k_0} Y_{k_0}\right), \tag{5}$$

where $\{t(j) : j = 1, \ldots, \Delta_i\} = M_i$ (as multisets).

For each $i \in [t]$ and $k = 0, \ldots, d-1$,

$$b_{i,k} - a_{i,k} Y_k > 0, \qquad 0 \le a_{i,k} \le M \qquad 0 \le b_{i,k} \le M. \tag{6}$$

where M is some (big) constant.

The proof of Theorem 3 is included in the full version of this paper [13].

Consider the branching matrix $\mathbf{M}_\ell$ generating the family of trees avoiding cycles of length $\le \ell$. Recall that the tree $T_{\mathbf{M}_\ell}$ which is generated by $\mathbf{M}_\ell$ is a super-tree of $T_{\text{saw}}(\mathbb{Z}^2)$. We show that the system (5)-(6) corresponding to $\mathbf{M}_\ell$ is feasible, proving SSM for $T_{\mathbf{M}_\ell}$ and hence for $T_{\text{saw}}(\mathbb{Z}^2)$.

To solve the feasibility problem, we add a new variable v in the right hand side of each linear constraint $ax \le b$, changing this constraint to $ax - b \le v$. We minimize v, which is an upper bound for the maximum violation by x among all constraints. The original linear system is feasible if and only the linear program has optimal solution $v < 0$.

The number of constraints and variables in this LP are huge (almost 10 billion constraints and 1 million variables) when $d = 200$ for the matrix $\mathbf{M}_8$. In order

to solve the linear program efficiently, one has significantly to reduce its size. In Section 6, we will discuss about the methods we use to solve this LP. When running the linear programs built for $\mathbf{M}_4$ we obtain $\lambda > 2.31$, and for $\mathbf{M}_6$ we obtain $\lambda > 2.45$, and for $\mathbf{M}_8$ we obtain $\lambda > 2.48$. In this way, we are able to prove that SSM holds for $\mathbb{Z}^2$ for $\lambda \leq 2.48$. The data for these LP solutions are available in our online appendix [16].

What we obtain from our linear program method are closer to the limit of this approach. Computational experiments suggest the threshold for WSM for $T_{\mathbf{M}_4}$ is at roughly $\lambda \approx 2.482$, for $T_{\mathbf{M}_6}$ at $\lambda \approx 2.653$, for $T_{\mathbf{M}_8}$ at $\lambda \approx 2.75$ and finally for $T_{\mathbf{M}_{10}}$ at $\lambda \approx 2.82$. These are thesholds for WSM, and the SSM threshold may in fact be even lower, as occurred for our example in Section 4.3.

5.1 Comparison with Previous Approaches

This method has several advantages compared to the method that is proposed in [11] in which a sufficient condition called the DMS condition, is introduced. DMS is a nonlinear matrix inequality obtained by comparing the geometric mean with the arithmetic mean when one analyzes the functional inequality (2) for a specific type of Ψ_i functions. These functions are the optimal ones when the tree $T_{\mathbf{M}}$ is a complete regular tree. However, for multi-type branching matrices, they are not necessarily optimal. One has to find the parameters of these functions Ψ_i in order to satisfy the DMS condition. The parameters for the DMS condition are obtained by a randomized hill-climbing program which may become trapped in a local optima. In contrast, the linear programming method we present here provides the optimal solution for the class of functions being considered.

For the SSM threshold of $T_{\mathbf{M}_\ell}$, our method includes the approximation of a more general class of functions and hence we obtain better lower bounds (see Figure 1 on page 710). Finally, the mathematical correctness of the linear programming method is very straightforward to check as compared to checking the correctness of the DMS condition. For $\ell = 4, 6, 8$, we summarize in the following table, the experimental lower bound for the WSM threshold of $\mathbf{M}_\ell$, the size of the matrix $\mathbf{M}_\ell$, the lower bounds of the SSM threshold for $\mathbf{M}_\ell$ obtained from DMS condition in [11], and the lower bounds of the SSM threshold for $\mathbf{M}_\ell$ obtained from our linear program approach.

ℓ	WSM threshold	Number of Types	λ from DMS in [11]	λ from LP
4	2.48	17	2.16	2.31
6	2.65	132	2.33	2.45
8	2.75	922	2.38	2.48

6 Reducing the Size of the LP

Initially, when we write down the linear programs (LPs) for the $\mathbf{M}_8$ matrix with the size of intervals around 10^{-3}, the number of constraints and variables is huge, approximately 10 billion constraints and 1 million variables. Solving this

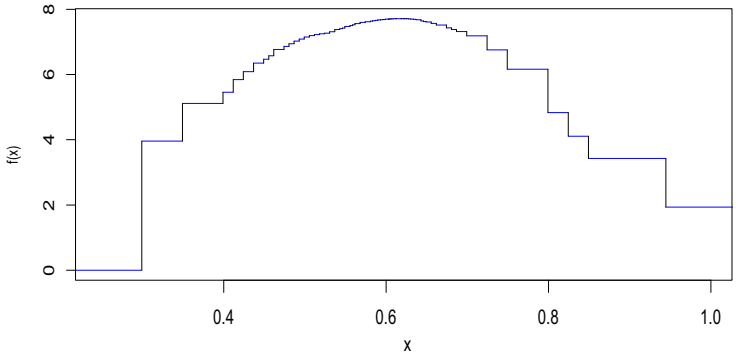


Fig. 1. A step function of Φ found by the LPs

LP directly is not possible, as the data will not even fit in memory. Notice that the LPs we create have high constraint-variable ratios. One standard technique to solve such LPs is to write the dual which has a high variable-constraint ratio and apply the column generation method [3]. From the primal point of view, we try to guess the set of tight constraints, by picking a set of primal constraints, solving a smaller LP and checking whether the rest of the constraints are satisfied. When there are violated constraints, several of the most violated constraints are added to the set and we iterate the procedure until the LP is solved.

Using column generation we obtain an LP that can be solved, but running the method takes too long. Next we will present two of our major techniques to reduce the size of the LPs so that we can solve them within a few days.

6.1 Nonhomogeneous Interval Size

In Theorem 3, we break the intervals into subintervals of the same size. The algorithm was designed to start with a very coarse set of the subintervals with a uniform length and if the LP has no solution, then the algorithm will try to decrease the length and re-solve the new LP. Usually, the algorithm has to make the length as small as 10^{-3} for the LP to have a solution. This creates lots of constraints. Notice that, the constraints are tight only in a very small range of the interval $(\frac{1}{1+\lambda}, 1)$. Therefore, we can try to break the intervals into subintervals of different sizes.

The goal of breaking intervals is to change the primal constraints so that the objective function v can be achieved at a smaller value. In column generation, shadow prices are used for this purpose. However, here deciding which intervals to break, affects the objective in a nonlinear fashion. Thus, we use a heuristic pricing scheme on the intervals to pick which ones to break. The following briefly describes our heuristic approach.

For each interval, we know how many constraints are involved for that interval and how many of them are violated (i.e., $ax - b > 0$). We sum up the values of $ax - b$ for how much each constraint is violated and then scale this by a factor of

the size of the interval to obtain what we define as its price. The algorithm will pick several intervals with the highest prices to break. The reason why we scale by a factor which is a function of the size of the interval is that we do not want to break the intervals that are already very small. In Figure 1 (see page 710) we show a step function Φ_i for a type i in $\boldsymbol{D}_8$ found by the LPs. One can observe from the figure that most of the intervals have large lengths; in fact, there are some small intervals in the middle as these are the intervals that create tight constraints.

6.2 Reduction of the Branching Matrices $\mathbf{M}_\ell$

Usually, when one applies various methods trying to solve the functional inequality (2), one has to face the fact that the dimension of the matrix $\mathbf{M}$ is huge, e.g., $t = 922$ for $\ell = 8$ in [11]. A natural way to generate $\mathbf{M}$ is using a DFS program that enumerates all of the types by remembering the history of the self-avoiding walk. However, there are many types in such a matrix that are essentially the "same". Here we provide a heuristic and rigorous method for finding those types that are the same.

Let $\mathcal{C}$ be a partition of the types in $\mathbf{M}$, i.e., $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ such that $\biguplus_{i=1}^{k} C_i = [t]$. We define the partition to be consistent with $\mathbf{M}$, if for every $i \in [k]$, each pair of types $s, t \in C_i$, the rows $\mathbf{M}_s$ and $\mathbf{M}_t$ are the same with respect to $\mathcal{C}$, that is

$$\sum_{j \in C_{i'}} \mathbf{M}_{sj} = \sum_{j \in C_{i'}} \mathbf{M}_{tj}, \text{ for all } i' \in [k].$$

Definition 4. *Given* $\mathbf{M}$ *and a partition* $\mathcal{C}$ *of size* k *which is consistent, we define the* k*-by-*k *matrix* $\mathbf{M}^{\mathcal{C}}$ *by,*

$$\mathbf{M}^{\mathcal{C}}_{ii'} = \sum_{j \in C_{i'}} \mathbf{M}_{sj} \text{ where } s \in C_i.$$

We say $\mathbf{M}$ *is* reducible *to a* k*-by-*k *matrix* $\mathbf{B}$ *if there is a consistent partition* $\mathcal{C}$ *such that* $\mathbf{B} = \mathbf{M}^{\mathcal{C}}$.

Lemma 8. *For a partition* $\mathcal{C}$ *of size* k*,*

$$\mathcal{F}_{\leq \mathbf{M}^{\mathcal{C}}} = \mathcal{F}_{\leq \mathbf{M}} \quad \text{and} \quad T_{\mathbf{M}^{\mathcal{C}}} = T_{\mathbf{M}}.$$

Proof. The argument is just a standard induction on the height of the tree. □

Now the question is how to find a good partition $\mathcal{C}$ easily. For a specific value $\lambda < \mathrm{WSM}(T_{\mathbf{M}})$, let V_λ be the fixed points of the recurrences of the marginal distributions defined by $\mathbf{M}$. Our conjecture is the following.

Conjecture 1. Let the partition $\mathcal{C}(\lambda)$ be the sets of types that have the same value of the fixed points in V_λ, i.e., for each $C_i \in \mathcal{C}(\lambda)$, for all $c \in C_i$, $V_\lambda(c)$ are the same. If for all λ, the partitions $\mathcal{C}(\lambda)$ are identical, then $\mathcal{C}$ is a partition that is consistent of $\mathbf{M}$.

Using the intuition from Conjecture 1 we are able to find good partitions in practice. We simply run a dynamic programming algorithm on the tree $T_{\mathbf{M}}$ to calculate an approximation of the fixed points in V_λ. Once the approximation is good enough, we simply make the partition according to this approximation. We then check the consistency of the partition with $\mathbf{M}$, and therefore, we know whether the resulting matrix generates the same tree as the original one or not by Lemma 8. Applying this reduction to $\mathbf{M}_6$ the number of types goes down from 132 to 34, and for $\mathbf{M}_8$ the number of types goes down from 922 to 162. This significant reduction in the size of the matrices greatly reduces the number of constraints and variables in our linear programming formulation. We use this technique to simplify the branching matrix $\boldsymbol{D}_G$ considered in Section 4.2 for proving Theorem 1, and reduce the matrix from 7 types to 3 types.

7 Conclusions

Current techniques for proving lower bounds on $\lambda_c(\mathbb{Z}^2)$ analyze SSM on $T_{\text{saw}}(\mathbb{Z}^2)$. This paper shows that this approach will not be sufficient to reach the conjectured threshold of 3.79.... One problem in this approach is that boundary conditions obtainable on $T_{\text{saw}}(\mathbb{Z}^2)$ are not necessary realizable on $\mathbb{Z}^2$. Some of the boundary conditions are more "extremal" than the one that is on $\mathbb{Z}^2$ which yields a lower weak spatial mixing threshold. Finding a way to exclude certain boundary conditions for $T_{\text{saw}}(\mathbb{Z}^2)$ would be an extremely interesting direction.

References

1. Bauer, F.L., Fike, C.T.: Norms and exclusion theorems. Numerische Mathematik 2, 137–141 (1960)
2. Baxter, R.J., Enting, I.G., Tsang, S.K.: Hard-square lattice gas. Journal of Statistical Physics 22(4), 465–489 (1980)
3. Bazaraa, M.S., Jarvis, J.J., Sherali, H.D.: Linear Programming and Network Flows, 3rd edn. John Wiley & Sons (2011)
4. van den Berg, J., Steif, J.E.: Percolation and the hard-core lattice gas model. Stochastic Processes and their Applications 49(2), 179–197 (1994)
5. Blanca, A., Galvin, D., Randall, D., Tetali, P.: Phase Coexistence and Slow Mixing for the Hard-Core Model on $\mathbb{Z}^2$, Preprint available from the arXiv at: `http://arxiv.org/abs/1211.6182`
6. Brightwell, G.R., Häggström, O., Winkler, P.: Nonmonotonic behavior in hard-core and Widom-Rowlinson models. Journal of Statistical Physics 94(3), 415–435 (1999)
7. Dobrushin, R.L.: The problem of uniqueness of a Gibbsian random field and the problem of phase transitions. Functional Analysis and its Applications 2(4), 302–312 (1968)
8. Gaunt, D.S., Fisher, M.E.: Hard-Sphere Lattice Gases. I. Plane-Square Lattice. Journal of Chemical Physics 43(8), 2840–2863 (1965)
9. Kelly, F.P.: Loss Networks. Annals of Applied Probability 1(3), 319–378 (1991)

10. Rácz, Z.: Phase boundary of Ising antiferromagnets near $H = H_c$ and $T = 0$: Results from hard-core lattice gas calculations. Physical Review B 21(9), 4012–4016 (1980)
11. Restrepo, R., Shin, J., Tetali, P., Vigoda, E., Yang, L.: Improved Mixing Condition on the Grid for Counting and Sampling Independent Sets. Probability Theory and Related Fields 156(1-2), 75–99 (2012)
12. Robinson, C.: Dynamical Systems: Stability, Symbolic Dynamics, and Chaos, 2nd edn. CRC Press, Boca Raton (1999)
13. Vera, J.C., Vigoda, E., Yang, L.: Improved Bounds on the Phase Transition for the Hard-Core Model in 2-Dimensions, Preprint available from the arXiv at: http://arxiv.org/abs/1306.0431
14. Watkins, D.S.: The matrix eigenvalue problem: GR and Krylov Subspace methods. SIAM (2007)
15. Weitz, D.: Counting independent sets up to the tree threshold. In: 38th Annual ACM Symposium on Theory of Computing (STOC), pp. 140–149 (2006)
16. Data is available from: http://www.cc.gatech.edu/~vigoda/hardcore.html

Author Index

Abbe, Emmanuel 332
Ahn, Kook Jin 1
Alaei, Saeed 11
Austrin, Per 26

Batman, Lucia 347
Beimel, Amos 363
Blanca, Antonio 379
Blum, Avrim 395
Braverman, Vladimir 42, 58

Campagna, Andrea 411
Chakrabarty, Deeparnab 71, 425
Chen, Sixia 436
Cohen, Edith 81, 452
Coudron, Matthew 468

Dani, Varsha 484
David, Roee 497
Diaz, Josep 484
Dumitrescu, Adrian 96
Dvir, Zeev 513

Edwards, Katherine 110

Feige, Uriel 497
Fekete, Sándor P. 126
Forbes, Michael A. 527
Fox, Kyle 142
Fraigniaud, Pierre 158
Friggstad, Zachary 173

Galvin, David 379
Goldhirsh, Yonatan 543
Griffiths, Simon 110
Grigorescu, Elena 559
Guha, Sudipto 1, 189
Guo, Alan 411
Guo, Zeyu 575
Gupta, Anupam 205
Guruswami, Venkatesan 591

Hajiaghayi, MohammadTaghi 11, 218
Halldórsson, Magnús M. 158
Haramaty, Elad 671
Hayes, Thomas 484
Hoffmann, Hella-Franziska 126
Hu, Guangda 513
Huang, Sangxia 233

Im, Sungjin 142
Impagliazzo, Russell 347
Ishai, Yuval 607

Kale, Satyen 205
Kaplan, Haim 81, 452
Kennedy, William Sean 110
Khandekar, Rohit 218
Khani, M. Reza 218
Kortsarz, Guy 218
Krishnaswamy, Ravishankar 71
Kulkarni, Janardhan 142

Li, Shi 71
Li, Yi 623
Liaghat, Vahid 11
Lu, Pinyan 639

Makarychev, Yury 244
Manokaran, Rajsekar 26
Mansour, Yishay 81, 260
McGregor, Andrew 1, 274
Meyerson, Adam 287
Montanari, Andrea 332
Moore, Cristopher 436, 484
Moseley, Benjamin 142
Munagala, Kamesh 189
Murray, Cody 347

Nagarajan, Viswanath 205
Narayanan, Srivatsan 71, 591
Nayyeri, Amir 244
Nissim, Kobbi 363

Ostrovsky, Rafail 42, 58
Oveis Gharan, Shayan 303

Pan, Feng 317
Patt-Shamir, Boaz 158
Paturi, Ramamohan 347

Randall, Dana 379
Rawitz, Dror 158
Reingold, Omer 655
Ron-Zewi, Noga 671
Rosén, Adi 158
Roth, Aaron 395
Roytman, Alan 287
Rubinfeld, Ronitt 411
Russell, Alexander 436

Sahai, Amit 607
Saket, Rishi 205
Scheder, Dominik 683
Schieber, Baruch 205
Schild, Aaron 317
Seshadhri, C. 425
Shpilka, Amir 527
Sidiropoulos, Anastasios 244
Steinke, Thomas 655
Stemmer, Uri 363
Stubbs, Daniel 274
Sudan, Madhu 671

Tagiku, Brian 287
Tan, Li-Yang 683
Tetali, Prasad 379
Tóth, Csaba D. 96
Trevisan, Luca 303

Vadhan, Salil 655
Vardi, Shai 260
Vera, Juan C. 699
Viderman, Michael 543, 607
Vidick, Thomas 468
Vigoda, Eric 699

Weiss, Mor 607
Wenner, Cenny 26
Wimmer, Karl 559
Woodruff, David P. 623

Xie, Ning 559

Yang, Linji 699
Yin, Yitong 639
Yuen, Henry 468